

RICE UNIVERSITY

**Coordination and Interference in 802.11 Networks:
Inference, Analysis and Mitigation**

by

Eugenio Magistretti

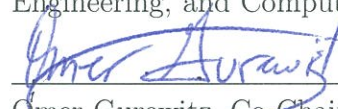
A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

APPROVED, THESIS COMMITTEE:



Edward W. Knightly, Chair
Professor of Electrical and Computer
Engineering, and Computer Science



Omer Gurewitz, Co-Chair
Assistant Professor of Computer Systems
Engineering, Ben Gurion University



David B. Johnson
Professor of Computer Science, and
Electrical and Computer Engineering



Ashutosh Sabharwal
Professor of Electrical and Computer
Engineering

Houston, Texas
December, 2012

ABSTRACT

Coordination and Interference in 802.11 Networks: Inference, Analysis and Mitigation

by

Eugenio Magistretti

In the last decade, 802.11 wireless devices data rates have increased by three orders of magnitude. However, network throughput, i.e., the successful bitrate at the MAC layer, has not seen a commensurate increase, and throughput imbalances still affect network links. This is a fundamental problem of wireless networks that is difficult to diagnose and amend. My research addresses two key causes of throughput loss and imbalance: MAC layer protocol overhead and destructive link interference. First, I design WiFi-Nano, a protocol that permits to reduce the channel access overhead by an order of magnitude, leveraging an innovative speculative technique to transmit preambles. This new concept is based on simultaneous preamble transmission and detection via a self-interference cancellation design, and paves the way to the realization of the collision detection paradigm in wireless networks. Next, I propose 802.11ec (Encoded Control), the first 802.11-based protocol that eliminates the overhead of control packets. Instead, 802.11ec coordinates node transmissions via a set of predefined pseudo-noise codewords, resulting in the dramatic increase of throughput and

communication robustness. Finally, I design MIDAS, a model-driven network management tool that identifies key corrective actions to alleviate underserved wireless links. MIDAS' key contribution is to reveal the fundamental role of node transmission coordination in characterizing destructive interference. I implement WiFi-Nano, 802.11ec, and MIDAS using a combination of WARP FPGA-based radio boards, custom emulation platforms, and network simulators. The results obtained show that WiFi-Nano increases the network throughput by up to 100%, 802.11ec improves network access fairness by up to 90%, and MIDAS identifies corrective actions with a prediction error as low as 20%.

In memory of my beloved Dad Pier Luigi.

Acknowledgments

Now that this journey is approaching the end, it is right to look back and acknowledge the numerous persons that contributed to the achievement. Of course, my first thanks go to my advisor, Prof. Edward Knightly, for his precious guidance in forming my research attitude and methodology. He taught me how to think clearly, rigorously, and originally; for all this, I am highly indebted to him. Immediately next, my most grateful thanks go to Prof. Omer Gurewitz, who has been like a true elder brother in directing and accompanying my research; his daily challenges represented a great gym for my critical thinking. I am grateful to my thesis committee members, Prof. Dave Johnson, and Prof. Ashu Sabharwal, who helped with constructive criticism to improve the clarity of my thesis and strengthen it.

I feel really fortunate to have the chance to share this journey with the members of the Rice Networks Group during these years, including Adriana, Ahmed, Bruno, Cen, Dee, Ehsan, Jasmin, Jingpu, Joe, Joshua, Marco, Misko, Naren, Oscar, Paul, Ryan, Tasos, Vincenzo. A big thanks go to the “physical layer” people at Rice, whom I have often bothered with hundreds of questions, and from whom I have learned invaluable practical lessons: Patrick, Achal, Melissa, Evan, Chris, Pedro, Sam, David, Clayton. I’d like to thank other ECE friends for the moments of fun we spent together: Florin, Keyvan, Michael, Predrag.

More thanks go to people I have shared special moments with, starting from the fellows of the Catholic Student Association: Sister Kitty, Father Bin, Father Justin, Tony, Andie and Landon, Sarah, Mike and Susan, Russell and Allison, Cristina. A big help in feeling at home came from Eveline; many thanks to her. Big thanks go to Taso, who was always there to discuss joys and concerns of our journey over dinner pizzas. A completely unique gratitude goes to Misko, the fellow of each and every day, who tolerated my exuberances and helped sharing my troubles; a beautiful person who has always been there for me. I owe large part of my achievements to wonderful persons and pedagogues who taught me the love of knowledge, and without whom all this would not have been possible: Prof. Ravaglia, Prof. Zappitello, Prof. Bellavista and beloved Prof. Paganini and Prof. Gentili.

I want to extend my gratitude to my Family, to my Sisters Cristina and Irene, and to my lovely niece Lucilla: she is the joy of my days in Bologna. Thanks also go to Gabriele and Davide. Lovely thanks go to my Grand Parents Maria, and the beloved Eugenio, Carlo, and Mimma, and my numerous Aunties, Uncles, and Cousins. Sometimes, also Friends become part of the Family; I wish to thank them in particular for alleviating my homesickness with their constant presence: Marco, Miles, Mirko, Gigi, Davide, Marco, Marco, Massimo, Paolo, Matteo, Gabriele, the Imolesi, the fellow engineers, and the beloved Gabri and Claudio.

The last thanks are very, very special. How could have I done all this without

my fiancée Natalia and her unconditional love and patience? She is my sunshine, my only sunshine! Mom Marisa and Dad Pier Luigi: only now have I come to realize the sacrifices they did to allow me follow my dreams. I am so sorry we had to be separated for so long. To Mom and Dad my greatest respect, love, and gratitude!

Finally, thanks be to the Trine God, the First and Only.

You are holy, Lord, the only God, and Your deeds are wonderful.

Contents

Abstract	ii
Acknowledgments	v
List of Figures	xii
1 Introduction	1
1.1 Thesis Contributions	7
1.2 Thesis Organization	10
2 WiFi-Nano	12
2.1 Introduction	12
2.2 Motivation and Background	17
2.2.1 802.11 Overhead Components.	17
2.2.2 802.11 Overhead Quantification.	20
2.2.3 Analysis of 802.11 9 μs Slot Length.	22
2.3 WiFi-Nano Overview	23
2.3.1 Contention Resolution	27
2.3.2 Fair Channel Access	32
2.3.3 Speculative ACKs	36
2.4 Preamble Detection in WiFi-Nano	36

2.4.1	Background on Carrier Sensing and Signal Correlation.	38
2.4.2	The Lattice Correlator	41
2.5	Simulation Results	43
2.5.1	Simulation Settings	43
2.5.2	Preamble Length in WiFi-Nano	44
2.5.3	Benefits of WiFi-Nano	49
2.5.4	Frame aggregation	53
3	802.11ec	55
3.1	MAC Protocol Design	60
3.1.1	Coded Control Information versus Message Control Information	61
3.1.2	11ec Channel Reservation Primitives	65
3.1.3	Discussion	68
3.1.4	Contending Flows and Vulnerability Interval	71
3.2	Experimental Evaluation of CSS	75
3.2.1	Experimental Setup	76
3.2.2	Channel Emulator Validation	77
3.2.3	CSS Length versus Robustness Trade-off	79
3.2.4	CSS Detection versus Control Message Decoding	81
3.2.5	11ec Codebook Size	84
3.2.6	Discussion on Signal Correlation	86

3.3	Experimental Evaluation of 11ec	89
3.3.1	Measurement-driven Network Emulator	90
3.3.2	Basic Topologies	92
3.3.3	Network Wide Emulation	101
3.3.4	Large Network Simulation	103
3.3.5	Extensions: Control During Data Simulation	104
3.A	Signal Correlation	107
4	MIDAS	109
4.1	Introduction	109
4.2	The MIDAS Framework	114
4.2.1	The Activity Share: Fundamental Element of Network Obser- vation	116
4.2.2	The Measurements	120
4.3	The Inference Tool	122
4.3.1	Network Model	122
4.3.2	Report-based Constraints	123
4.3.3	Entropy-based Statistical Solution	126
4.3.4	Protocol-based State Space Reduction	130
4.4	Mitigation of Hinderer Transmissions	131
4.4.1	Evolution of the Activity Share after Rate-limiting	132

4.4.2	Relationship between the Collision Probability of the Under-Served Link and the Activity Share	134
4.5	Performance Evaluation	138
4.5.1	Experimental Testbed	138
4.5.2	RSSI-based Busy Time Discovery	141
4.5.3	Testbed Results	143
4.5.4	Simulation Results - Inference Tool	153
4.5.5	Simulation Results - Throughput Prediction Tool	168
4.5.6	Simulation Results - Real Network Topology	173
5	Related Work	178
5.1	WiFi-Nano	178
5.2	802.11ec	181
5.3	MIDAS	182
6	Conclusions	184
	References	188

List of Figures

2.1	802.11 Packet Transmission Timeline at 600 Mbps.	18
2.2	802.11 Overhead Components at Different Data-rates.	21
2.3	WiFi-Nano Packet Transmission Timeline at 600 Mbps.	24
2.4	Making slot duration independent of carrier sensing time.	26
2.5	Chained Deferral Property of WiFi-Nano.	29
2.6	The Near Far Problem	30
2.7	Design of slot width in WiFi-Nano	35
2.8	Carrier Sensing Using Pseudo-Random Preamble	39
2.9	The Lattice Correlator	42
2.10	Preamble length for different detection thresholds	46
2.11	Number of contending transmitters over time	48
2.12	Throughput	50
2.13	WiFi-Nano overheads	51
2.14	Fairness	52
2.15	Frame Aggregation	54
3.1	Timeline of a packet exchange with Coded Control versus Message Control.	62

3.2	Timeline of the vulnerability interval of 802.11ec (time indications are in μs).	73
3.3	Example of 127-symbol CSS correlation at -6 dB.	79
3.4	Robustness vs length tradeoff for different CSS lengths.	81
3.5	Probability of missing CSS detection vs missing message decoding. . .	84
3.6	Low cross-correlation of CSS's different from the one transmitted. . .	86
3.7	Layout of my office building deployment.	92
3.8	Throughput of 11ec, 802.11 with/without RTS/CTS in basic topologies	96
3.9	Total airtime utilization in the case of Asymmetric Hidden Terminals.	98
3.10	Throughput distribution for a 5-flow topology.	102
3.11	Throughput of 11ec, 802.11 with/without RTS/CTS in 20-node simulated topologies	105
3.12	Effects of the use of the feature of Control during Data in 11ec on Information Asymmetry topologies.	107
4.1	Optional caption for list of figures	118
4.2	Layout of my testbed deployment.	141
4.3	Effect of the RSSI threshold on the busy time of node a	144
4.4	Activity Share inference (testbed).	145
4.5	Inference sensitivity to network density (testbed).	148
4.6	Inference sensitivity to traffic load (testbed).	150

4.7	Inference sensitivity to short report intervals (testbed).	152
4.8	Throughput increase estimation for concurrent nodes with loads in [400 kbps, 900 kbps] (testbed).	154
4.9	Activity Share: simulation vs testbed.	156
4.10	Inference with protocol-based reduction.	158
4.11	Inference sensitivity to density (10 Nodes).	160
4.12	Inference sensitivity to density (15 Nodes).	161
4.13	Sensitivity of the Activity Share estimation to traffic load (10 Nodes).	162
4.14	Inference robustness to missing reports.	164
4.15	Inference robustness to realistic traffic.	166
4.16	Inference robustness to short report intervals.	167
4.17	Comparison with an Exponential Inference Method	169
4.18	Throughput increase estimation for scenarios with density 3.	171
4.19	Throughput prediction sensitivity to density (600 kbps).	173
4.20	Throughput prediction sensitivity to density (900 kbps).	174
4.21	Topology of the TFA network deployed in Houston, Texas.	175
4.22	Activity Share Inference for the TFA network (fully backlogged nodes).	176
4.23	Throughput Increase Prediction for the TFA network (transmission rate 600 kbps).	177

Chapter 1

Introduction

The presence of wireless networks in people's daily lives is becoming ubiquitous and, correspondingly, people's expectations and demands are growing more and more ambitious. This has established a virtuous cycle driven by applications that are increasingly advanced, and likewise bandwidth hungry. Over the course of the last fifteen years, IEEE 802.11 has emerged as a major enabling technology of high-speed wireless networks, by providing enhanced performance leveraging crucial innovations in the communications field. OFDM, MIMO, MU-MIMO, radio design evolution supporting higher order modulations and wider bandwidth have altogether contributed to increase the achievable data rates from 11 Mbps of 802.11b, to the anticipated Gbps-range of 802.11ac/ad.

Despite these efforts, 802.11 throughput, i.e., the successful bitrate at the MAC layer, has not seen a commensurate increase, and throughput imbalances still affect network links. In fact, throughput loss and imbalance is a fundamental problem of wireless networks, whose origin is often hard to determine. In this thesis, I focus on two root causes, namely MAC layer protocol overhead and destructive link interference.

A large part of 802.11 standards is devoted to PHY and MAC layer specifications:

the former prescribe how the information is conveyed over the air interface via the transmission of physical signals, while the latter address the issue of inter-link interference by determining at each time instant which links have the right to access the channel. While 802.11 PHY layer specifications have significantly evolved to support two orders of magnitude higher datarates, the MAC layer and in particular the channel access strategies have remained largely unchanged from the original version. However, while the channel access overhead was relatively small for the long packet durations at the original low datarates, it becomes a major source of inefficiency for the short transmissions at the new high rates. Two major contributors to the channel access overhead are backoff slot size and control messages. Unfortunately, both involve physical layer operations that cannot just be shortened leveraging the communication techniques used to increase the datarate.

The second cause of throughput loss I study in this thesis is link interference in the form of packet corruption. This occurs when carrier sensing fails to coordinate the transmissions of conflicting links, usually because of topological conditions, e.g., in the presence of hidden terminals. In such situation, at least one of the involved links observes high throughput losses. 802.11 has historically tackled this issue by introducing optional RTS (Request-To-Send) and CTS (Clear-To-Send) control messages for the reservation of the medium. The downside of enabling RTS/CTS is that, since they need to be received by all conflicting links, they require to be transmitted at

low modulation rates and thus significantly increase the protocol overhead. This introduces an unavoidable trade-off between overhead and effective channel reservation that ends up penalizing the network throughput.

Finally, packet corruption is not the only source of link interference and thus cause of link under-performance. In 802.11 networks, barring perfectly symmetric link connectivity, also carrier sensing may be responsible for imbalanced throughput. These two causes may combine in different and complex ways in real network deployments and propagate their effects across multiple nodes and hops. In essence, their effect depends on a multiplicity of elements such as topology, traffic load, and transmission pattern. Because of this complexity, it is extremely difficult to characterize the interactions and interference between different links. Consequently, even the design of corrective actions to improve the under-served links becomes a daunting task, since the effect of such actions is largely unpredictable.

1.0.0.1 WiFi-Nano: Near-Zero Channel Access Delay

Slot duration is a major contributor of 802.11 channel access overhead, since it determines the transmission delay due to backoff. According to 802.11, the slot duration should allow a station to determine whether other stations access the channel in the current slot via preamble detection and, in the negative case, to prepare the radio for their own transmission at the beginning of the subsequent slot. The duration of these

two operations is governed by physical layer phenomena, and thus cannot be shortened without sacrificing their correctness and finally incurring in packet collisions. Thus, 802.11 slot duration is the minimum feasible value.

I argue that this conclusion hinges on the assumption that preamble detection and transmission are serial operations. My fundamental observation is that if the stations can detect preambles while simultaneously transmitting their own preambles/packets, the slot size can be significantly reduced. Based on this observation, I design a novel PHY/MAC protocol, *WiFi-Nano*, with one order of magnitude shorter slots than 802.11 and, correspondingly, one order of magnitude shorter channel access delay. The key of my design is the speculative preamble transmission technique, where nodes continue to detect preambles while transmitting, leveraging recent physical layer enhancements [55].

1.0.0.2 802.11ec: Collision Avoidance without Control Messages

The exchange of MAC control information is essential for the correct and effective operations of MAC protocols: for example, RTS/CTS mitigate interference and ACKs confirm correctly received data. Even though the amount of information exchanged is small, the overhead of exchanging such information via MAC messages can be significant, as in addition to the control information they need to include preamble, source/destination address, message type, etc., all of which need be transmitted at

base rate in order to guarantee robust delivery. For example, an ACK message is 14 byte plus physical layer encapsulation, but contains only one bit of relevant information (that data was successfully received). Likewise, RTS/CTS is rarely used in practice precisely due to excessive overhead despite its decisive role in mitigating collisions.

In my thesis, I design, implement, and evaluate 802.11ec (Encoded Control) as a control message free MAC. Instead of control messages, I propose to employ correlatable symbol sequences (CSS's), predefined binary codewords that can be detected via correlation at the receiver instead of decoded. My key observation is that, together with their transmission timing, CSS's have the potential to convey all control information, thus changing the fundamental design properties of the MAC. In fact, the characteristics of CSS's provide the unique opportunity to reduce the overhead of control information by an order of magnitude, while even increasing its robustness to potential interference by enabling its reception at low SINR. Unfortunately, the CSS length strikes a balance between overhead - the shorter the better; and robustness and dictionary size - the longer the better. In order to maintain the overhead low, I show how to solve the challenge of representing all 802.11 control information with a limited number of CSS's.

1.0.0.3 MIDAS: Measurement-driven Modeling for Online Throughput Prediction

Regardless of the actual protocol enhancements implemented, the management of 802.11 wireless networks demands the acquisition, representation, and predictable variation of the network operating point. In practice, a fundamental task of network management is to improve under-served links, and a key corrective action to achieve such goal is throttling other nodes that may be hindering the target links. However, to do so first requires identifying which nodes are hindering the target links, i.e., the acquisition and representation of the operating point, and how much to throttle them, i.e., the predictable variation of the operating point. Unfortunately, as I show in my thesis, this task is extremely challenging. In fact, there may be a large set of candidate nodes for which throttling can have vastly different effects on a target link, e.g., my results show throughput benefits ranging from as little as 7% to as much as 172% of the rate limited quantity. Furthermore, the benefit of any throttling action is hard to predict because of the complex node interactions.

In this thesis, I design MIDAS, a framework that uses online measurements of network performance to infer the most hindering nodes which cause a target link to be underserved, and to identify effective rate limiting strategies to predictively increase a target link's throughput. My key original contribution is to reveal the fundamental importance of node transmission coordination to characterize destruc-

tive interference. Capturing coordination is extremely challenging as it consists of measuring and quantifying an elusive and complex process, i.e., how links influence each others' transmission behaviors. In order to capture coordination, I originally introduce the concept of Activity Share, defined as the fraction of time a set of nodes spend transmitting simultaneously. The Activity Share permits assessment of current networks conditions; however, it lacks predictive power to identify effective actions and to anticipate their outcomes. Thus, I design the first throughput prediction model that uses online measurements, i.e., the inferred Activity Share.

1.1 Thesis Contributions

WiFi-Nano, 802.11ec, and MIDAS address with original solutions and designs the fundamental problems of low throughput links in 802.11 networks, focusing on two causes, namely MAC layer protocol overhead and destructive link interference. Specifically, my thesis makes the following contributions:

- **WiFi-Nano:** *Speculative Transmissions for Channel Access Delay Reduction.*

Slot duration and channel access delay reduction require a completely new approach to the physical layer operations they encompass. Speculative transmissions introduce the new concept of simultaneous preamble transmission and detection and pave the way to the realization of the collision detection paradigm in wireless network. Speculative transmissions enable a new design where the slot

time can be arbitrarily shortened without penalizing the protocol performance.

- **WiFi-Nano:** *Interference Reduction via Collision Detection and Fairness Enhancement via Counter Roll-back.*

The implementation of speculative transmissions requires addressing two fundamental design issues related to fairness, i.e., guaranteeing that all nodes in a single contention domain have identical transmission opportunities, and to contention resolution. The spatial bias resulting from the interaction between shorter slot size, propagation delay, and signal attenuation highly challenge these goals. The determination of slot and preamble length, as influenced by propagation delay and node to node spreading of contention resolution, and the design of novel backoff counting rules such as the roll-back, permit to guarantee fair channel access and to enable collision detection almost eliminating the interference due to simultaneous backoff expiration.

- **802.11ec:** *Efficient and Robust Delivery of MAC Control Information via CSS's.*

CSS's radically transform the way MAC control is conveyed. In fact, CSS's reduce control overhead by nearly an order of magnitude, while increasing control robustness thanks to small duration, which entails a lower probability of suffering interference, and high resilience to low SINR. The design of novel map-

ping techniques leveraging the locality principle and timing codes allow to use a limited number of CSS's, i.e., to fully benefit from their advantageous properties, while substantially preserving the information content of control messages intact.

- **MIDAS:** *The Activity Share: a Fundamental Element of Network Observation.*

MIDAS' key original contribution is to unveil the fundamental importance of node transmission coordination to characterize link interactions, and in particular interference relationships. The Activity Share permits to uniquely measure coordination because it represents the essence of node interaction, i.e., the simultaneous or sequential relationships between the transmissions of different nodes. Unfortunately, nodes cannot locally measure the Activity Share; thus, MIDAS comprises an inference tool that estimates it using a small set of passively collected measurements.

- **MIDAS:** *The First Online Measurement-based Throughput Prediction Tool.*

The ultimate goal and contribution of MIDAS is to provide a management tool that permits to design corrective actions aiming to increase the throughput of targeted under-served links. The Activity Share represents the current network operating point, capturing nodes coordination; however, coordination strongly depends on the exact traffic load and pattern of the nodes. This prevents the

Activity Share to be directly used for predicting the evolution of the operating point following the execution of corrective actions, and calls for the design of a throughput prediction tool. MIDAS includes the first prediction model that, differently from all related work (see Section 5), obtains its input from a representation of the present state of the network.

- **Overall:** *System Implementation and Evaluation Showing Large Gains.*

I extensively evaluate the three solutions designed and show their large performance gains through first of their kind prototypes. In particular, I implement and verify the systems using the Rice University’s Wireless Open-Access Research Platform (WARP) FPGA-based platform, via a combination of software and digital design. However, real experimentation setups and field trials are often practically limited in scale; thus, I complement my approach via large scale measurement-driven emulations and simulations. Example performance figures are: almost 100% throughput gain for WiFi-Nano, over 20 fold throughput gain for 802.11ec, and a prediction error below 20% for MIDAS.

1.2 Thesis Organization

The thesis is organized as follows. In Chapter 2, I introduce WiFi-Nano starting from an in-depth analysis of 802.11 channel access overhead. Successively, I present WiFi-Nano’s technical issues and solutions, including key elements to enhance channel

access efficiency and fairness. Finally, I show simulation results delineating the large achievable gains. In Chapter 3, I propose 802.11ec and discuss the complete protocol design emphasizing the technical insights needed to map control information to a limited number of CSS's. The final part of the chapter presents the evaluation results consisting of a thorough investigation of CSS performance in a controlled environment, and of a campaign of emulations and simulations confirming the benefits of adopting CSS in small as well as large scale networks. In Chapter 4, I describe MIDAS; the chapter includes an introduction to the Activity Share, followed by the technical presentation of the inference and throughput prediction tools. Finally, experimental and simulation results that MIDAS accurately infers the Activity Share and assesses alternative corrective actions. In Chapter 5, I cover the most important related work, by classifying it according to the three main directions of my work. In Chapter 6, I conclude by discussing the implications and future directions of the novel research paths delineated in this thesis work.

Chapter 2

WiFi-Nano

2.1 Introduction

While WiFi physical layer (PHY) data rates have increased from 1 Mbps in the original 802.11 to 1 Gbps in the upcoming 802.11ac standard, user level throughputs have not seen a commensurate increase. A key reason for this is the *channel access overhead*. As I show in Section 2.2, the average channel access delay is $16 \mu s + 9.5 * slottime$. Since 802.11a/n slots are $9 \mu s$, the average channel access delay is $101.5 \mu s$. Thus, while the transmission time for a 1500 byte packet is only $20 \mu s$ at 600 Mbps, the average channel access overhead is over 500% of the packet transmission time.

Given that the slot size plays a crucial role in WiFi's inefficiency, a fundamental question emerges: is it feasible to reduce the slot to less than $9 \mu s$? Consider a typical backoff slot. When the backoff counter expires at one node, the node starts transmitting its packet. Preambles in 802.11 are transmitted at the beginning of each packet and contain predefined sequences that help the receiver detect the packet reliably. As I discuss in Section 2.3, the $9 \mu s$ slot is designed to accommodate $4 \mu s$ needed for packet detection/clear channel assessment (CCA) [64] and about $5 \mu s$ of turnaround time, designed to accommodate propagation delay, processing and time for switching the radio from receive to transmit. In order to reduce the slot time

below $9\ \mu\text{s}$, one would have to either reduce the time for turnaround or the CCA time. However, in the former case, after CCA determines that the carrier is idle, nodes may be unable to transmit on the next slot given lack of sufficient turnaround time; in the latter case, collisions would increase due to missed detection of busy carrier.

While the above observations confirm that the *802.11a/n slot size of $9\ \mu\text{s}$ is close to the minimum feasible value*, I argue that it is based on one key assumption – preamble transmission and detection is serial, i.e., one device transmits a preamble at any given time while others are performing CCA. Instead, if preamble transmission and detection could be done in parallel, i.e., if devices could detect preambles that are being transmitted from other devices while simultaneously transmitting their own preambles/packets, then $9\ \mu\text{s}$ slots are superfluous and slot sizes can be significantly reduced. Based on this crucial original observation, my solution to reducing the channel access overhead is a novel PHY/MAC design, *WiFi-Nano, with slots as small as $800\ \text{ns}$* .

My key novel contribution is the *speculative transmission* technique. In WiFi-Nano, all transmitters *speculatively transmit their preambles* in the slot where their backoff counters expire. Since preamble detection (CCA) can now take multiple slots, i.e., up to 5 ($4\ \mu\text{s} / 800\ \text{ns}$), devices continue to detect preambles even while transmitting their own preambles – nodes accomplish this by using analog self-interference

cancellation [55], which allows nodes to remove the effect of their own transmission before performing preamble detection on the received signal. If a device detects a preamble from another device before it finishes transmitting its own preamble, then it aborts its transmission immediately since this implies that the other device had initiated its transmission earlier. Thus, average channel access time can be reduced to $7.6 \mu s$ ($9.5 \times 800 \text{ ns}$), an order of magnitude lower than $101.5 \mu s$ in WiFi.

In order to enable the gains of speculative preamble transmissions, I need to address two challenges related to node channel access fairness and collision avoidance. First, nodes may unduly count down their backoffs while other nodes' preambles are being transmitted. In fact, until the moment a preamble is detected, which may occur as late as 5 slots after the preamble transmission begins, the channel is perceived free. In order to re-establish the correct backoff values, once a preamble is detected, nodes roll back their counters, aligning the counters with the beginning of the preamble transmissions. This rollback technique requires that each node accurately estimates the starting time of any detected preamble (Section 2.4). In this thesis, I propose and design the *lattice correlator*, an original detector that permits to identify sub-parts of a pseudo-random preamble and, combined with a novel preamble structure, to accurately determine the preamble starting time. Second, speculative preamble transmissions may allow several nodes to simultaneously transmit preambles, and thus generate a high interference. This may finally result in preamble mis-detections and

thus cause collisions. The speculative preamble technique permits to address this issue with minimal penalty. In fact, the transmission deferral enjoys an avalanche effect property that gets triggered during channel access. Specifically, when several devices speculatively transmit, nodes in their vicinity detect these high SNR preambles and immediately abort their respective transmissions, thereby quickly reducing overall interference. Combined with the flexibility in the design of the novel WiFi-Nano design this permits to tackle the issue by slightly enlarging the preamble duration. My results show that a preamble only about $4\ \mu\text{s}$ longer than in WiFi permits to achieve almost perfect detection.

Performing preamble detection in parallel with preamble transmission permits to enable two unique benefits unprecedented in wireless networks. First, packet collisions due to identical choice of backoff counters by multiple nodes can be almost eliminated. If two or more devices initiate their preambles in the same slot, these devices will be able to detect the start time of each other's preamble and deduce that collision is imminent. In such a case, they probabilistically abort the transmission of their preambles such that, with high likelihood, only one node remains during the transmission of the payload of the packet. Second, since backoff counter collisions are unlikely in WiFi-Nano, unfairness due to capture-effect between near-and-far terminals [36] is also eliminated.

Apart from channel access overhead, the speculative transmission technique can

also be used to reduce the acknowledgement (ACK) overhead in WiFi. While an 802.11n ACK packet consumes negligible transmission time at 600 Mbps, a $40\ \mu\text{s}$ preamble, needed to correctly deliver an ACK packet, coupled with $16\ \mu\text{s}$ SIFS implies that the ACK overhead is 2.8X ($56\ \mu\text{s} / 20\ \mu\text{s}$) the transmission time of a 1500 byte packet. In WiFi-Nano, instead of waiting for SIFS before transmitting the ACK preamble, the receiver speculatively starts transmitting its ACK preamble as soon as it finishes receiving a data packet. While the preamble is being transmitted, the node finishes decoding the packet. Upon detecting error in reception, the receiver simply aborts its ACK transmission. This *speculative ACK preamble transmission* allows WiFi-Nano to eliminate SIFS and, thus, reduce the ACK overhead.

The realization and implementation of WiFi-Nano requires analog self-interference cancellation [55] and the ability to transmit and receive simultaneously. The former is an inexpensive noise canceller circuit while the latter requires an extra oscillator and antenna. An extensive set of simulations (Section 2.5) show that WiFi-Nano highly improves the efficiency of WiFi.

In summary, I make the following contributions:

- The design and implementation of WiFi-Nano, a novel PHY/MAC design that improves the throughput of WiFi by up to 100% using the following techniques.
- The speculative preamble transmission technique that permits to limit the slot length to 800 ns, and thus to reduce the channel access overhead by an order

of magnitude.

- The interference reduction via collision detection and fairness enhancement via counter roll-back that allow to nearly eliminate same slot interference and provide all nodes with equal opportunities to access the channel.
- The speculative ACK transmission that permits to eliminate the ACK overhead due to SIFS.

2.2 Motivation and Background

In this section, I motivate the design of WiFi-Nano by analyzing the overhead present in 802.11. First, I separate the overhead in four components, and examine each one. Then, I quantify the impact of each component as evidenced by a set of simulation results. Finally, I conclude the section with an in-depth analysis of 802.11 9 *mus* slot duration.

2.2.1 802.11 Overhead Components.

First, I consider a single device transmitting 1500 byte data packets back-to-back using 802.11n at 600 Mbps using the Distributed Coordination Function (DCF) with RTS/ CTS turned off – a common choice for today’s deployments. The transmission time for a 1500 byte (maximum MTU allowed by Ethernet) packet at 600 Mbps is 20 μs . This is accompanied by three key overhead elements, namely, *channel access*,

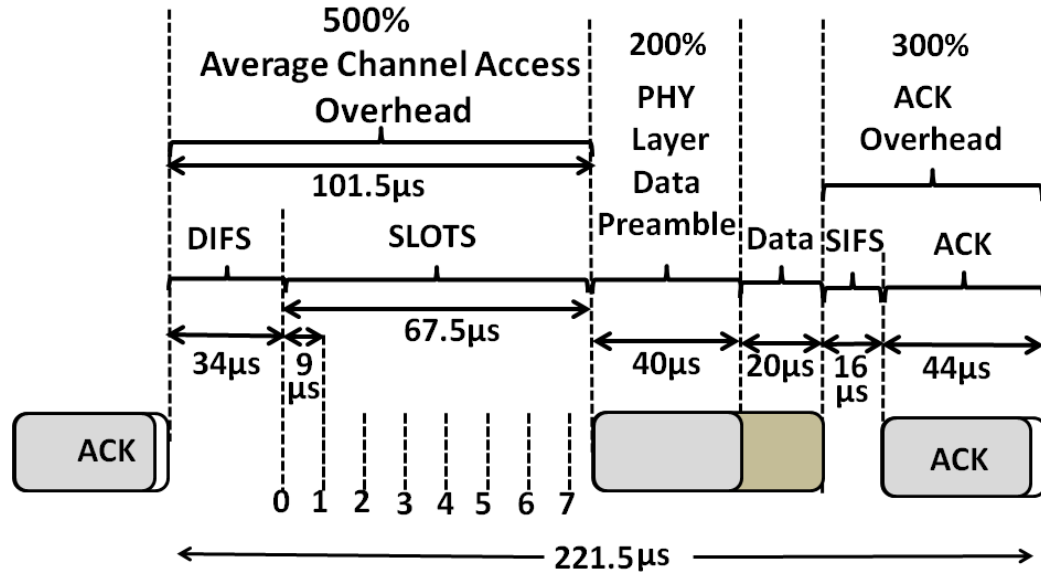


Figure 2.1 802.11 Packet Transmission Timeline at 600 Mbps.

data preamble and *acknowledgement* as shown in Figure 2.1.

Channel Access Overhead. As dictated by CSMA in 802.11, prior to transmitting its next packet, a device must first sense that the channel is idle for the duration of DIFS. DIFS, which is $34\mu s$ long, comprises SIFS ($16\mu s$) and 2 slots (each $9\mu s$). After DIFS, a node must typically defer its transmission for a random number of slots, generated from 0 to CW (contention window size) to allow other devices to share the channel in a fair manner. Given that the minimum value of CW as dictated by 802.11 is 15, the device will, on average, wait about 7.5 slots before transmission. Thus, the average overhead for channel access amounts to $16\mu s + 9.5 * \text{slots}$, i.e., $101.5\mu s$, and is independent of the transmission data rate (Figure 2.1). At 600 Mbps, it is about

500% of the data transmission time.

Data Preamble Overhead. The transmission of data in every packet is preceded by a physical layer preamble. The preamble is crucial in preparing the receiver for a successful reception. First, it helps the receiver to reliably establish that a packet is being transmitted (*packet detection*) and detect the boundaries of various parts of the packet (*synchronization*) to enable decoding. Second, it helps in *channel estimation* *i.e.*, helps combat the vagaries of the wireless environment by providing sufficient information to allow the receiver to estimate and correct for the channel characteristics. Third, for 802.11n MIMO receptions, it helps the receiver to estimate the MIMO parameters required to allow leveraging the spatial orthogonality of multiple streams. Thus, while 802.11a/g preambles are 20 μs long (including PLCP header), for 4x4 802.11n, preambles are 40 μs long – 200% of packet transmission time (Figure 2.1).

Acknowledgement Overhead. Upon the successful reception of a packet, the receiver responds with an ACK. In order to allow enough time for the receiver to process incoming data and prepare its radio for transmission, nodes must wait for SIFS (16 μs) before transmitting an ACK. The ACK content itself comprises only 14 bytes and should take only 18 ns at 600 Mbps. However, since 802.11n uses 4 μs symbols, all packets transmission durations must span multiples of 4 μs . Thus, 802.11 pads the ACK packet with zeros to make it a 4 μs worth of data. Further, the ACK also includes a preamble that is 40 μs . Thus, as depicted in Figure 2.1, SIFS and

ACK together span $60\ \mu\text{s} - 300\%$ of packet transmission time.

So far, I have described overheads in the context of a single device. When several devices contend for the channel, packet collisions occur resulting in one more element, *i.e.*, the *collision overhead*.

Collision Overhead. When multiple devices contend, their backoff counters are decremented independently and in parallel. The wait time for accessing the channel is thus determined by the device with the minimum backoff counter value. As a result, overhead (idle time) due to channel access reduces as the number of contending devices increase. However, with increasing contention, the probability that two or more devices may choose to transmit in the same slot increases, leading to increased collisions. Each collision then results in a wasted time interval equal to the sum of data transmission time, data preamble time and ACK transmission time.

2.2.2 802.11 Overhead Quantification.

Figure 2.2 depicts the fraction of air time due to different overhead components for a 1500-byte data transmission at three different data rates (54, 300, and 600 Mbps) in two deployment scenarios.* The first scenario includes a single transmitter having unequivocal access to the channel, and is representative of a sparse topology; the second, comprising 30 transmitters all randomly located within carrier sensing range

* These are results obtained using the Qualnet network simulator, with settings as in Section 2.5.

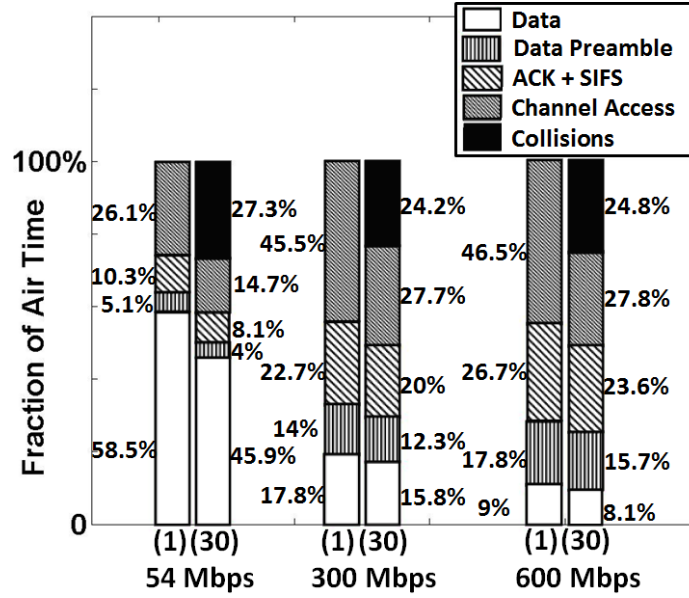


Figure 2.2 802.11 Overhead Components at Different Data-rates.

of each other, represents a dense deployment environment. As data rates increase from 54 Mbps to 600 Mbps, the fraction of time occupied by data transmissions, i.e., the efficiency of the system, reduces from about 58% to 9%. The sum of data preamble and ACK overhead increases from 15.4% at 54 Mbps to 44.5% at 600 Mbps. Finally, the sum of channel access and collision overhead amounts to 26.1% (42% for 30 transmitters) to as high as 46.5% (52.6% for 30 transmitters).

Given that preamble and ACK are indispensable, these overheads are not amenable to significant reductions. *The primary focus of this chapter is thus to reduce the overhead due to channel access and collisions, and improve the efficiency of WiFi.*

2.2.3 Analysis of 802.11 9 μs Slot Length.

As described in the standard, 802.11 9 μs slot length includes four delay components accounting for physical layer operations that need to be performed within a single slot; these are *carrier sensing*, *Rx-Tx turnaround*, *signal propagation* and *MAC processing*. While 802.11 recommends nominal durations for each of these components, individual manufacturers have some degree of flexibility in determining the delays based on their specific hardware capabilities and constraints. However, the sum of these delays must amount to less than 9 μs .

Carrier Sensing. Before initiating its own transmission in the current slot, a device must reliably establish that no other transmission was initiated in the past slot. A failure to do so will result in packet collision. As described in Section 2.4, given the vagaries of noise and interference, the time required to ascertain the presence/absence of an ongoing transmission depends on the signal to interference ratio (SINR) at the receiver. 802.11 recommends 4 μs to enable reliable carrier sensing from the farthest nodes in the network.

Rx-Tx Turnaround. Given that WiFi devices are not required to allow simultaneous transmission and reception, several components, *e.g.*, antenna and RF oscillator, are shared between the transmission and reception circuits. Thus, devices require time to switch from reception to transmission mode in order to reset these shared components. 802.11 recommends 600 ns as the switching time.

Signal Propagation. RF waves travel a distance of 100 m in approximately 330 ns. Thus, if two devices are 100 m apart then their carrier sensing and notion of slot boundaries may be 330 ns apart. Slots must accommodate these delay effects due to propagation delays. 802.11 recommends a value of 800 ns to accommodate for speed of light.

MAC Processing. Each signal from PHY layer needs to be processed by the MAC layer and then the MAC must issue signals to the PHY. This turn around time depends on specific hardware implementations.

2.3 WiFi-Nano Overview

Three fundamental elements differentiate WiFi-Nano from 802.11.

- *800 ns Slots.* Instead of using 9 μs slots, WiFi-Nano uses 800 ns slots – more than an order of magnitude reduction in slot duration.
- *Speculative Preamble Transmission.* Devices begin speculatively transmitting their preambles when their respective backoff counters expire, even before channel access contention has been resolved. Contention for channel access is carried out simultaneously while preambles are being transmitted, aided by analog self-interference cancellation [55]. All devices, except the ones whose backoff counters expired the earliest, abort their transmissions mid-way (devices B and C in Figure 2.3).

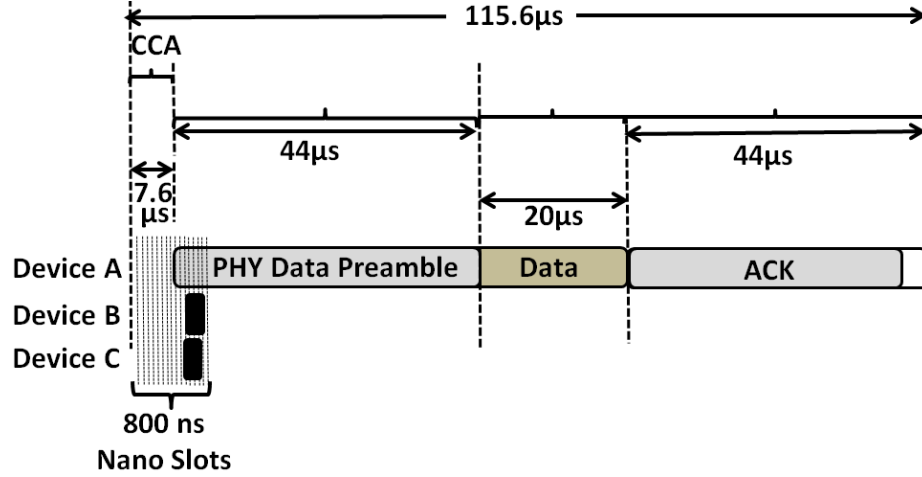


Figure 2.3 WiFi-Nano Packet Transmission Timeline at 600 Mbps.

- *Speculative ACK.* WiFi-Nano devices eliminate the need for SIFS by speculatively transmitting the preamble even as the received packet is being processed. The ACK transmission is then aborted mid-way upon detecting errors in the received packet.

Figures 2.3 and 2.1 visually represent the timeline of WiFi-Nano and WiFi, respectively, operating at 600 Mbps. As a result of these changes, WiFi-Nano *dramatically reduces channel access delay* given that the delay is function of SIFS and slot duration (Section 2.2). Furthermore, as carrier sensing is carried out while transmitting preambles, devices headed for a packet collision are able to detect this condition with high probability and resolve their contention probabilistically, resulting in a *near zero collision probability*. Finally, speculative ACKs bring about a reduction in ACK overhead by up to 35%.

In the remainder of the section, I present the key ideas and innovations in the design of WiFi-Nano. First, I cover the fundamental contribution of WiFi-Nano, i.e., the speculative preamble technique, by describing an illustrative example. Then, I discuss the properties, challenges, and solutions addressing the issues of fairness and contention resolution. I conclude the section introducing the speculative ACK technique.

Speculative Preamble Transmission. WiFi-Nano removes the dependency of slot duration on carrier sensing time by requiring devices to carrier sense while speculatively transmitting data preambles. To illustrate this idea, I consider a simple example that adopts 800 ns slots in Figure 2.4. Two WiFi-Nano devices A and B contend for the same channel. For the sake of simplicity, in this example we assume that the whole $4\ \mu s$ recommended by the standard are required by the devices to detect each others transmissions. As seen from Figure 2.4, device A finishes counting down its backoff and initiates its transmission before B. Device B finishes its countdown one slot (800 ns) after A. Since B requires $4\ \mu s$ to detect an ongoing transmission, B is unable to detect A’s transmission at this time. However, instead of waiting, Node B speculatively initiates its own transmission. Given that in WiFi-Nano B can carrier sense while transmitting, B eventually detects A’s transmission four slots later. B determines that the other node (A) started transmitting earlier than itself, since B started transmitting less than $4\ \mu s$ before. Consequently, B concludes that it cannot

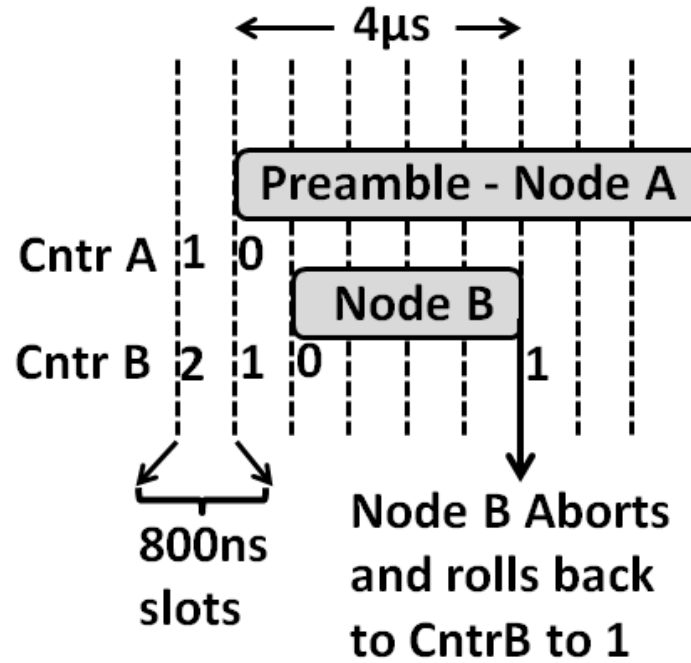


Figure 2.4 Making slot duration independent of carrier sensing time.

be the rightful owner of the medium and aborts its transmission mid-way. Device A on the other hand continues its own transmission uninterrupted. *Thus, by enabling carrier sensing while transmitting and using speculative transmission of preambles, channel contention can be performed simultaneously to preamble transmissions, hence “masking” channel access overhead under the necessary preamble overhead. Further, slot duration is no longer constrained to be less than the carrier sensing duration.*

The interaction between shorter slot size, propagation delay, and signal attenuation highly challenge the goals of limiting the node collisions, and of providing equal channel access opportunities to all nodes. Next, I introduce few issues related to these

two goals, and propose practical solutions that altogether compose the WiFi-Nano design.

2.3.1 Contention Resolution

Speculative preamble transmissions and short slot length encourage many nodes to simultaneously access the channel as soon as the medium is perceived as free. This may end up causing a large amount of interference and, in turn, hindering the contention resolution thus generating collisions. My design defines a simple deferral rule that triggers a domino effect resolving node contention. Finally, speculative preambles grant WiFi-Nano the unique capability to anticipate collisions. The probabilistic deferral technique leverages this capability and nearly eliminates collisions even due to same slot transmissions by different nodes.

Aborting Ongoing Transmissions Rule. Using the speculative preamble technique a node may detect a preamble, i.e., another node’s attempt to access the channel, while the node itself is transmitting. In that case, the transmitter needs to decide whether or not to abort its own transmission. The simple rule I suggest is based on the capability of the physical layer implementation to identify the starting time of a received preamble (see Section 2.4). Specifically, the transmitter aborts its current transmission only if another preamble’s reception began in an earlier slot (the case of same slot beginning is described below).

Chained Deferral Property. While the speculative preamble transmissions tend to increase the number of devices that access the channel simultaneously, the abortion rule I just introduced tend to reduce the interference. It is important to remark that the time needed to detect another node's preamble is a function of the preamble SINR, i.e., ultimately of the distance between the nodes (more details on this issue are provided below and in the next section). For this reason, proximal nodes' preambles are detected sooner than farther nodes' preambles. As proximate devices detect each other's preambles due to the high SINR and abort, the preambles of farther devices, which initially enjoy a much smaller SINR due to the interference of proximate transmissions, become amenable for detection because of the reduced amount of interference due to the abortion of proximate devices. Thus, as time progresses, more and more devices abort their transmissions in a chain reaction like manner, expanding in geographical extent. This domino effect is depicted in Figure 2.5, where at each stage, dark colored nodes are devices that are transmitting preambles. As seen from Figure 2.5 and evaluated in Section 2.5, the inherent parallelism of chained contention resolution in WiFi-Nano is extremely quick to resolve all contentions in the network. However, this still implies that WiFi-Nano may require larger preambles than WiFi.

An example of the deferral property is in Figure 2.6. In the depicted topology, devices B and C are proximate to each other, while device A is far from both B and

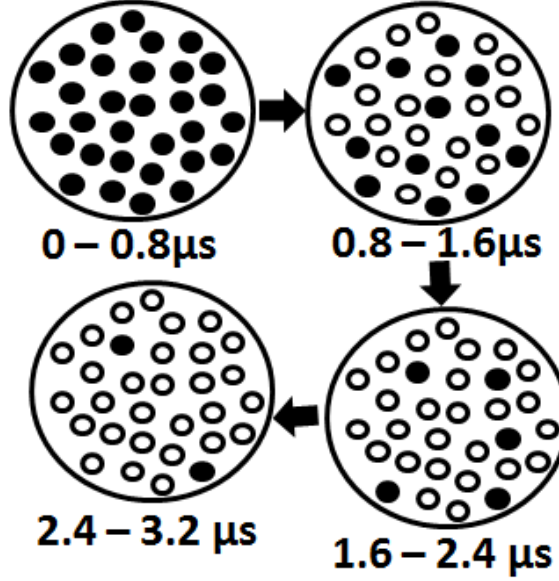


Figure 2.5 Chained Deferral Property of WiFi-Nano.

C. The signal strength of links $A \rightarrow B$ and $A \rightarrow C$ is -85 dBm, while the signal strength of link $B \rightarrow C$ is -45 dBm. In this example, I assume that device A started to speculatively transmit its preamble first followed by devices B and finally C. A's transmission at B will be overwhelmed by the transmission from C and will have an SINR of -40 dB ($= -85 \text{ dBm} - (-45 \text{ dBm})$); the same will happen at C. Suppose that an SINR of 10dB is required to reliably detect A's transmission. Then, in order for B or C to detect A reliably, the preamble transmitted by A must have processing gain of about 50 dB. As described in Section 2.4 the preamble length needed to achieve a processing gain of 50 dB is 100000, which translates to a preamble that is 5 *ms* long – an impractically long time. However, such a long preamble from A is necessary

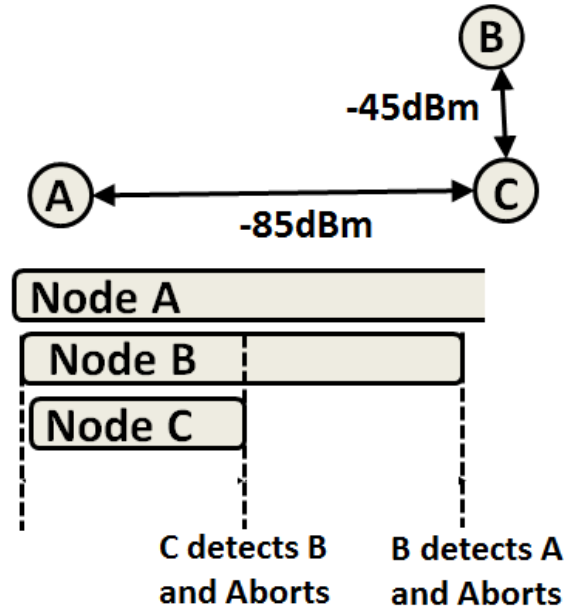


Figure 2.6 The Near Far Problem

only if B and C are continuously transmitting. Since B and C are very close to each other, aided by self-interference cancellation, C receives B with a very high SINR. Thus, C can detect B in a very short time and abort its transmission. Once C aborts its own transmission, B only needs 10 dB of processing gain to detect A's preamble; this enables B to detect A and also abort.

Probabilistic Same-Slot Deferral. I mentioned above that a transmitter may detect a preamble whose reception began exactly in the same slot as the local transmission. Same slot collisions are known to affect carrier sensing based MAC protocols, even in the most favorable operating conditions. Differently from all literature, WiFi-Nano's preamble detection capabilities allow to nearly eliminate even this type

of collisions; furthermore, this can be achieved without penalizing the channel access efficiency. In fact, in case two nodes begin transmitting in the same slot, WiFi-Nano interference cancellation permits both of them to detect the potential for a packet collision. However, requiring both nodes to abort their transmissions may hinder the channel efficiency by entirely wasting an otherwise almost completed contention. For this reason, WiFi-Nano permits to both nodes to continue the transmission for one more slot with a probability of 50%. Thus, in the next slot, with 50% probability one node wins sole access to the channel. However, with 25% probability, both nodes might abort in which case a new contention resolution phase can begin; finally, with 25% probability, both nodes might decide to continue transmitting. In the latter case, the probabilistic collision resolution process continues again in the next slot. Thus, eventually with high probability either both devices abort or only one continues transmitting, thereby winning sole access to the channel. WiFi-Nano preamble detection implementation enjoys one more beneficial property, i.e., it can approximately estimate the number of overlapping preambles received. This permits to identify the number of contending nodes that chose the same slot as the local transmitter and, thus, tune the abortion probability to such estimated value. Suppose that a node detect $k - 1$ distinct contending preamble transmission received in the same slot as the local transmission, that node continues transmitting with a probability of $\frac{1}{k}$. Thus, the probabilistic collision resolution mechanism in WiFi-Nano avoids payload colli-

sions with high probability, thereby significantly reducing the collision overhead seen in Figure 2.2.

2.3.2 Fair Channel Access

Speculative preamble transmissions may bias the channel access even for nodes located in a single contention domain where all nodes perfectly carrier sense each other. This is mainly due to two reasons, i.e., multi-slot carrier sensing and signal propagation delay. In order to solve these issues, WiFi-Nano design establishes a novel rule for backoff counter management and dictates a minimal slot length.

Backoff Counter Rollback. Because of the short slot duration, the detection of a received preamble may occur only several slots after the reception started. In the meantime, the backoff counter at the receiver may have been unduly advanced of several slots, i.e., of an interval corresponding the delay between reception and detection. This issue can be illustrated by continuing the example in Figure 2.4, where both devices A and B speculatively initiate their transmissions after the backoff expires hoping to gain access to the channel. Even though B rightfully aborts its transmission, B's backoff counter value has been unduly advanced of one slot more than it should have. This is because if B had not experienced any delay in carrier sensing, it would have suspended its backoff counter at 1, i.e., as soon as A initiated its transmission. Since the detection delay is not constant but depends on the SINR

between transmitter and detector, this phenomenon depends on the network topology. Furthermore, it may lead to unfair situations where a node systematically benefit of the unduly counter advancement more than other nodes.

In order to preserve fair access, I propose the counter rollback technique where any device detecting a preamble rolls back its backoff counter to the starting time of the transmission that wins a contention. Effectively, an aborting node adds to its current backoff value the number of slots elapsed between the starting and the detection time of the first preamble detected. Notice that all detecting nodes apply this technique, and not only the ones that are transmitting at the moment of detection. Finally, it is important to remark that the correct application of the counter rollback technique requires that a node continues to monitor the channel even after it detects a first preamble and determines the channel as busy. This is because the first preamble detected may not identify the winner of the contention. In fact, it may be possible that an earlier preamble transmission by a farther node is detected with a larger delay and thus the backoff counter needs to be updated via multiple rollbacks, i.e., until the earliest preamble that identifies the contention winner is detected. The exact implementation of the rollback mechanism will be described in greater detail in Section 2.4.

Minimum Length of WiFi-Nano Slots. Requiring the devices to perform carrier sensing while transmitting necessitates that the receiver and transmitter operate

completely independently. This requirement inherently eliminates the need for Rx-Tx switching delay as the receiver and transmitter are simultaneously operating at any given time. Furthermore, since MAC processing overheads only result in delayed speculative transmissions that may be aborted, their inclusion in slot duration is no longer crucial. Consequently, the only remaining contributor to the slot duration is the propagation delay. How does propagation delay affect the duration of a slot? As described earlier in this section, in order to preserve fairness, devices must rollback their backoff counters to the time of initiation of the transmission that grabs the channel. Propagation delays cause incorrect estimation of this time. *If slot durations are less than twice the maximum propagation delay of the network, these errors result in incorrect rollbacks resulting in unfairness.*

I illustrate this using a simple example. As depicted in Figure 2.3.2 two devices A and B are located very close to each other so that the propagation delay from A to B is close to zero. Device C however, is located at the edge of the network with a propagation delay of 400 ns from either A or B. Consequently, after A ends its transmission, while B realizes immediately that the channel is idle and starts counting down, C realizes this only after 400 ns. Suppose that node C has a countdown value of 0 so that it starts to transmit immediately at $t = 400 \text{ ns}$. This transmission will reach B after another 400 ns and hence B will start receiving C's transmission at $t = 800 \text{ ns}$. Consequently, when B detects C's transmission it will assume that C

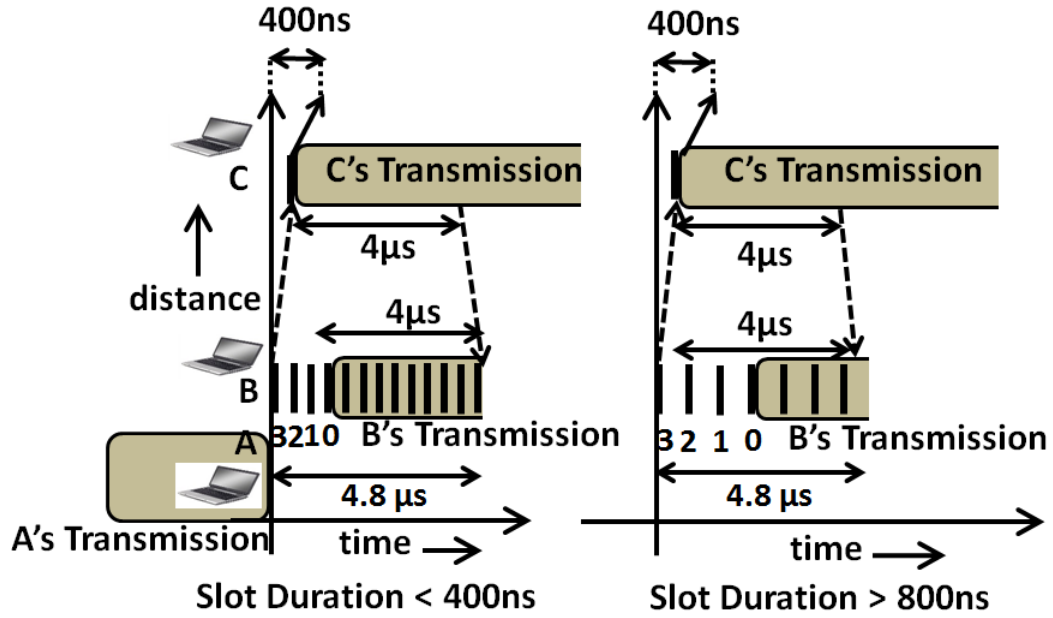


Figure 2.7 Design of slot width in WiFi-Nano

started at $t = 800 ns$ and roll its counter back to a time corresponding to $t = 800 ns$. As depicted in Figure 2.3.2, this error has no effect if the slot duration is greater than $800 ns$ *i.e.*, twice the propagation delay. Since I am targeting an indoor AP-based setting with a range of $100 m$ (propagation delay of $333 ns$), I use a slot width of $800 ns$ in WiFi-Nano. Note that, even with short $800 ns$ slots, WiFi-Nano will continue to work for networks with larger ranges, albeit with some level of unfairness depending on the network topology.

2.3.3 Speculative ACKs

WiFi-Nano also eliminates the need for SIFS between data and ACK transmissions. According to the 802.11 standard, SIFS of $16 \mu s$ is designed to accommodate delays such as transferring the received signal from the antenna, packet decoding, MAC processing delay, and time to switch from receive to transmit mode. Since WiFi-Nano nodes have separate transmit and receive paths, the receiver can simply start speculatively transmitting the ACK preamble as soon as reception is complete. In parallel, the node decodes the received packet and computes the CRC to check if there are any errors. Upon detection of any error, the node simply aborts the transmission of the ACK preamble. Since preamble length even at 802.11a rates is $20 \mu s$, there is ample time for processing and aborting ACKs. Finally, note that speculatively transmitting ACKs does not require full-duplex capability since there is no transmission during reception; I simply overlap the processing of already received data during transmission.

2.4 Preamble Detection in WiFi-Nano

In this section, I discuss the various components that allowed for the implementation of WiFi-Nano. Two challenges need to be addressed. First, speculative preambles require that a preamble is received even while a transmission is ongoing. In order to address this issue, WiFi-Nano leverages self-interference cancellation, according to the

architecture proposed in [55]. To avoid the device's transmissions from overwhelming its receiver, the signal from the transmitting antenna is subtracted from the received signal at the receiving antenna, thus mitigating the interference due to devices' own transmission. To increase robustness of detection, WiFi-Nano allows a longer carrier sensing time without affecting slot duration and hence the efficiency significantly.

The second challenge stems from two requirements of the preamble detection rules described in the last section. Chained deferral requires that devices detect other preambles as soon as possible. To achieve this, WiFi-Nano leverages the fact that the stronger the received transmission, the faster it can be carrier sensed. Consequently, while proximate devices create a strong interference, they also abort quickly allowing transmissions from weaker devices to be detected. The second requirement derives from the backoff rollback technique: in order to adjust the counter a device must also be able to accurately the time the preamble reception began. WiFi-Nano uses a novel preamble detector - *sub-preamble lattice correlator* - which allows detection of continuous sub-parts of the preamble. In this section, I focus on the original implementation of this later component. I start by providing a necessary background on the carrier sensing technique that 802.11 adopts, based on the detection of pseudo-random sequences.

2.4.1 Background on Carrier Sensing and Signal Correlation.

Carrier Sensing Using Pseudo Random Sequences. Wrongfully concluding an ongoing transmission when there is none, *i.e.*, false alarm, leads to loss of throughput as devices defer transmitting needlessly. On the other hand, missing legitimate transmissions from devices that may be far away as their received signal strength is weak, may lead to collisions. In order to allow reliable detection, in 802.11 standards nodes transmit pseudo-random noise (PN) sequences in the initial part of the preamble. Then, a receiver detects an ongoing transmission by correlating the received signal with the PN sequence. The key advantage of using a PN sequence is the sharp distinct peak that it provides exactly when the input signal to the correlator matches the PN sequence itself. In practice however, even though the transmitter transmits the exact PN sequence expected by the receiver, the received signal is effected by the wireless channel and seldom remains exactly the same. Consequently, the correlation of the PN sequence affected by the wireless channel may be poor.

Schmidl and Cox Correlator [62]. In order to combat the effects of the wireless channel, a more robust scheme by Schmidl [62] *et al.* transmits two or more copies of the same PN sequence (802.11a/n uses ten copies of the same 800ns sequences). Since each transmitted PN sequence copy is affected by the wireless channel the same way, the individual received copies at the receiver still remain identical. Thus, instead of correlating against the original PN sequence, received copies are correlated with

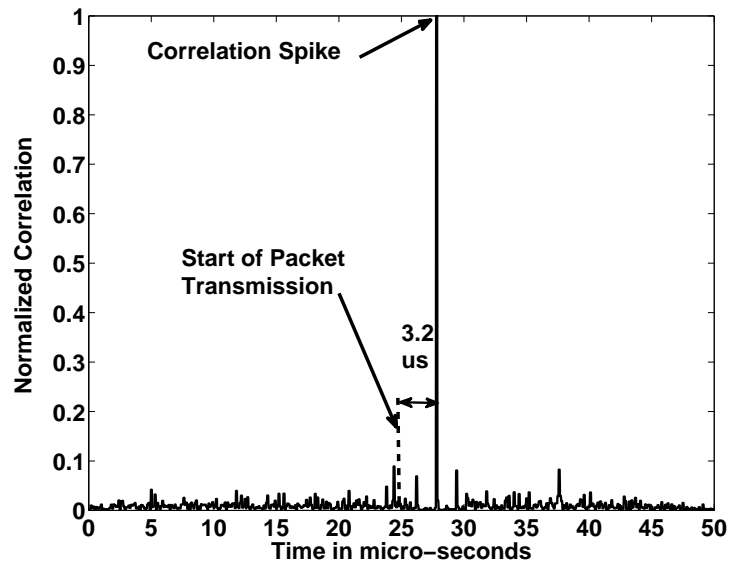


Figure 2.8 Carrier Sensing Using Pseudo-Random Preamble

each other. The key disadvantages of this scheme is that correlation peak is not as sharp as the standard PN sequence correlator and that the identification of the initial reception time is not immediate.

Tufvesson Correlator [67]. In my implementation, I favor a hybrid scheme that combines the advantages of the standard PN sequence correlation and Schmidl and Cox correlator. In this scheme, each copy is correlated with the known PN sequence and then outputs of the copies are correlated with each other to obtain a sharp and robust correlation spike. Figure 2.8 depicts the output of a Tufvesson correlator as a function of time to an incoming $3.2 \mu s$ pseudo-random preamble with two identical $1.6 \mu s$ copies. As seen in Figure 2.8, the correlator provides a sharp correlation spike

3.2 μs after the beginning of the reception of the preamble. *Thus, the time of transmission initiation can also be determined since the spike always occurs at the end of the reception of the entire preamble.*

Performance of PN based schemes. Correlation based detection with a pseudo-random sequence makes the detection more reliable by effectively increasing the signal to interference and noise ratio (SINR) of the correlation spike through constructive accumulation of signal energy and correlation. The correlation value over the noise floor determines the reliability of detection. Specifically, the correlation value depends on the received SINR of the preamble and the length of the preamble as $SINR \times \Delta$, where Δ is called the *processing gain* of the preamble. The processing gain of a preamble increases linearly with the length L of the sequence. For example at 20 MHz, since each sample is 50 ns long, a 4 μs long pseudo-random sequence will have an effective length of 80 (4000ns/50ns), *i.e.*, a processing gain of $10 \log(80) = 19 \text{ dB}$. If the received SNR of the preamble is 0 dB, then the correlation value of the PN sequence obtained at the receiver will be 19 dB over the noise floor, allowing a very reliable detection. For the interested reader, an in-depth discussion is provided in the next chapter (e.g., see appendix 3.A).

2.4.2 The Lattice Correlator

In order to take advantage of preamble abortion and guarantee chained deferral, and to correctly implement counter rollback, WiFi-Nano correlator and preamble structure must provide two functions. First, devices are required to correlate subparts of a preamble. For example, in Figure 2.6, after C aborts, B should be able to correlate only the part of A's preamble that was received after C aborted. This permits to achieve a higher SINR, hence a higher processing gain, on A's preamble. In general, when several nodes contend for the channel, the correlation may be required to be performed on different lengths and subparts of the preamble. Further, detection of subparts of the preamble allows the earliest possible detection and hence allows aborting the transmission at the earliest possible time. The second need emerges from the counter rollback technique and requires that the correlating nodes determine the exact position of the correlation within the preamble, since this permits to identify the beginning of the packet transmission.

In order to enable both these functionalities, WiFi-Nano design includes novel correlator and preamble structures as depicted in Figure 2.9. Each packet of WiFi-Nano is preceded by a PN sequence comprising several short but distinct 800 ns PN sequences PN1, PN2, \dots , PNn. The lattice correlator takes as input the received signal and generates $\frac{N(N-1)}{2}$, (N is the number of 800 ns PN sequences) outputs corresponding to the correlations obtained from each continuous sub-part of the preamble

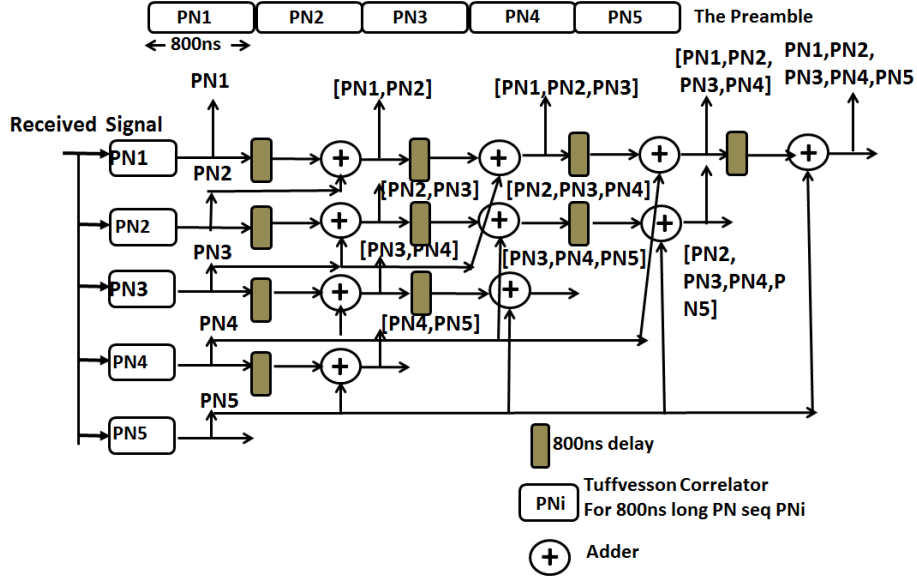


Figure 2.9 The Lattice Correlator

e.g., [PN1, PN2], [PN3, PN4, PN5] *etc.*. Detection of a spike in any of these inputs provides two pieces of information. First, the presence of an ongoing transmission and second, the time of the beginning of the reception. The time of beginning of the packet reception is determined by the position of the last 800 ns PN sequence. For example, a spike due to the correlation [PN2, PN3, PN4] indicates that the packet reception started $4 * 800 \text{ ns} = 3.2 \mu\text{s}$ before. While stronger transmissions are typically detected in the early stages of the lattice correlator, weaker signals may be detected at later stages.

2.5 Simulation Results

In this section, I extend my evaluation using simulations to determine the scalability of WiFi-Nano to larger deployments and its robustness to alternative parameter choices. Specifically, *i)* I investigate the choice of the preamble length, and derive values that permit a node to detect any transmitter in the network with high probability; *ii)* I evaluate the benefits of WiFi-Nano in terms of throughput and fairness, as compared to 802.11; *iii)* I investigate the effect of frame aggregation.

2.5.1 Simulation Settings

I implemented WiFi-Nano’s preamble detection physical layer in the Qualnet network simulator as an independent module that interfaces below the standard MAC 802.11. The module intercepts packet transmission requests from the MAC layer and performs the preamble contention phase before attempting the transmission. In the simulations, unless specified otherwise, I use a preamble of $8\ \mu s$ for packet detection in WiFi-Nano compared to $4\ \mu s$ in WiFi. The reason for this choice is explained in further detail in Section 2.5.2. To simulate the fact that self-interference is not perfect, driven by measurements obtained from my platform, I increased the noise floor by 5 dB whenever a device is transmitting.

I simulate 802.11a and 802.11n modulation rates; while typically the data-rate is fixed during a single experiment, I also present results with the auto-rate fallback

algorithm for 802.11a. All nodes emit a transmission power of 20 dBm. I choose a path-loss exponent of 2.72 so that the transmission range is 100 m for the 6 Mbps modulation; In all simulations, nodes are deployed such that they are able to carrier sense each other and, thus, there are no hidden nodes. Unless otherwise specified, the nodes generate fully backlogged CBR traffic, with packet size of 1480 bytes.

By default, WiFi-Nano reacts to preamble collisions using WiFi’s binary exponential backoff algorithm. However, I also explore the benefits of using IdleSense [22] for WiFi-Nano when there is high contention. Specifically, I simulate an idealized version of IdleSense with a fixed contention window of 350 when there are 30 active transmitters.

2.5.2 Preamble Length in WiFi-Nano

WiFi recommends about $4 \mu s$ of its preamble to perform carrier sense. In the rest of this section I shall refer to this as the *carrier sensing preamble length*. The rest of the WiFi preamble ($36 \mu s$ in 802.11n) is used for performing other functions such as channel estimation, synchronization and MIMO parameter estimation.

As discussed in Section 2.3, in order to avoid packet collisions while using speculative preamble transmissions, WiFi-Nano devices will require a longer preamble to reliably perform carrier sensing compared to WiFi devices. Thus, the carrier sensing preamble length in WiFi-Nano has to be longer than that used in WiFi. *More specif-*

*ically, the carrier sensing preamble in WiFi-Nano should be long enough to guarantee that all speculatively transmitting devices are able to reliably detect and abort their transmissions, allowing only the earliest device to transmit. Using a carrier sensing preamble of inadequate length will lead to packet collisions as more than one device will continue transmitting its packets. Since using a longer preamble adversely affects efficiency, in this section I ask the question, “What is the shortest possible carrier sensing preamble * length that WiFi-Nano can use while ensuring that preambles are correctly detected even in high contention scenarios?”*

A key parameter that dictates preamble length is the SINR required for reliable preamble detection without false alarms. While my testbed results indicate that an SINR detection threshold of about 8 dB is sufficient. in noisy environments higher values of SINR detection threshold may be required. The higher the SINR detection threshold, the longer the carrier sensing preamble length should be. Consequently, in this section I evaluate the carrier sensing preamble length for 10, 15 and 20 dB detection thresholds for considering extremely noisy environments. For each of these detection thresholds, I determine the minimum carrier sensing preamble length that limits the probability of missed preamble detection to under 1%. In my evaluations, in order to be conservative, I consider an extreme scenario for WiFi-Nano – 30 fully

*Note that the duration of a WiFi-Nano preamble will be the sum of its carrier sensing preamble length and the part of the preamble used in WiFi for other functions such as synchronization, channel estimation etc. For example, a WiFi-Nano carrier sensing preamble length of 8 μs would result in a total preamble length of 44 μs (36 μs + 8 μs) instead of 40 μs (36 μs + 4 μs) in 802.11n and 24 μs instead of 20 μs in 802.11a.

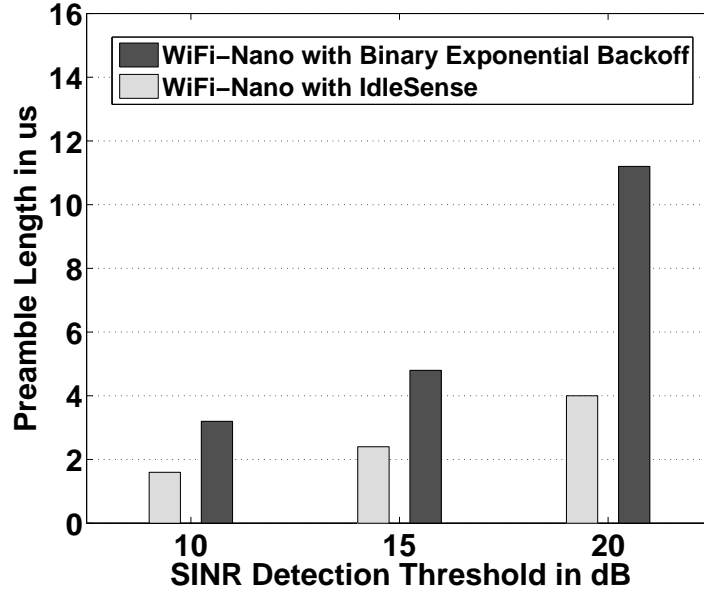


Figure 2.10 Preamble length for different detection thresholds

back-logged transmitters, deployed randomly in a 100 m diameter region. All results were computed as averages over five random topologies each running for a period of 30 s. I tried two backoff schemes – exponential backoff (used by WiFi) and IdleSense.

Figure 2.5.2 depicts the minimum carrier sensing preamble length as a function of SINR detection threshold. As seen from Figure 2.5.2, when using exponential backoff, a length of $4\ \mu s$, same as used by WiFi, is sufficient when the SINR detection threshold is 10 dB (corresponding to my testbed results). As SINR detection thresholds are increased to 15 dB and 20 dB, the required carrier sensing preamble length is $7.2\ \mu s$ (an increase of $3.2\ \mu s$ over WiFi) and $11.2\ \mu s$ (an increase of $7.2\ \mu s$ over WiFi) respectively.

Also notice that IdleSense has a beneficial effect for WiFi-Nano. By limiting the number of nodes contending in a single slot, IdleSense reduces interference and consequently results in a shorter carrier sensing preamble length. In particular, IdleSense is able to maintain the preamble length at $4 \mu s$ even in the noisy 20 dB case. In other words, *no change in preamble length will be required in WiFi-Nano while using IdleSense*. For the remaining results, I fix the detection threshold to 15 dB, and conservatively choose a preamble length of 2 OFDM symbols, i.e., $8 \mu s$.

In order to shed better insight as to why the preamble length does not increase dramatically in WiFi-Nano, I investigate the preamble detection process by evaluating how quickly nodes are able to carrier sense and abort their transmissions. For my evaluation, I divide each simulation into epochs – each epoch starts at the end of an ACK transmission for the last packet when the channel becomes idle. In each epoch I count the number of nodes that are transmitting at various increasing time intervals from the start of the epoch. The average number of active transmitting nodes as a function of time interval elapsed from the beginning of an epoch is computed by averaging over all epochs in the simulation.

Using the same 30 node setting as before, Figure 2.5.2 plots the average number of transmitters versus the time displacement after the beginning of the epoch. For instance, a point (1.6,1.8) means that after $1.6 \mu s$ since the channel became idle, 1.8 nodes on average are transmitting speculative preambles. The channel contention is

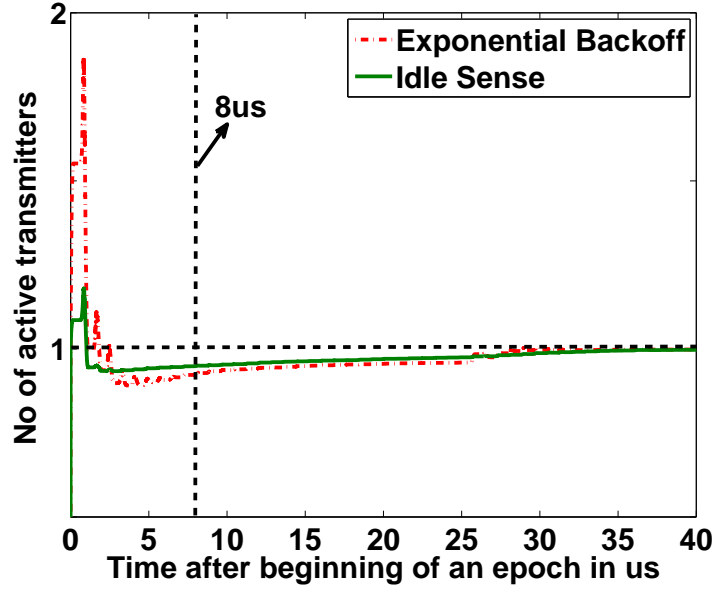


Figure 2.11 Number of contending transmitters over time

resolved when the average number of transmitting nodes is equal to 1. As seen in Figure 2.5.2, the average number of transmitters decreases rapidly as nodes abort upon detecting other transmitters. After a short period of less than $4 \mu s$, the average number of transmitters reduces to slightly below one. The average is below one because colliding transmitters abort probabilistically and, in rare cases, all transmitters may abort.

During the interval between $4 \mu s$ and $30 \mu s$, the curve shifts upward slowly representing a small increase in the average number of transmitters. This occurs because when all colliding transmitters have aborted, some nodes that did not detect these aborted transmissions had their backoff counters expire and thus they grabbed the

idle channel. Finally, the average number of transmitter converges to 1, indicating that only one transmitter grabs the channel and transmits successfully. This indicates that typically not more than 4 μs of carrier sensing preamble may be necessary for WiFi-Nano. Finally, the benefit of IdleSense is evident in this figure, as the number of contenders at the beginning of each epoch is dramatically reduced.

2.5.3 Benefits of WiFi-Nano

WiFi-Nano has three main benefits over 802.11. WiFi-Nano *i)* significantly reduces the overhead of data transmission, thus, increasing throughput even when one node is transmitting; *ii)* limits the collision overhead to the length of the preambles and thus improves throughput further when there are many contending nodes; *iii)* improves fairness by eliminating the capture effect, in which nodes closer to an AP, take advantage of the higher SINR with respect to farther nodes and gain higher throughput.

Throughput. Figure 2.5.3 compares the throughput achieved at different data-rates in two cases, when there is only one transmitter in the network and when there are 30 active transmitters in the cell. Consider the case of a single transmitter. As the data-rate increases, the relative improvement of WiFi-Nano over 802.11 increases, due to high channel access overhead of 802.11 at higher rates as described in Section 2.2. Thus, the throughput gain of WiFi-Nano over 802.11 is 5%, 37%, 85%, 88%, at 6

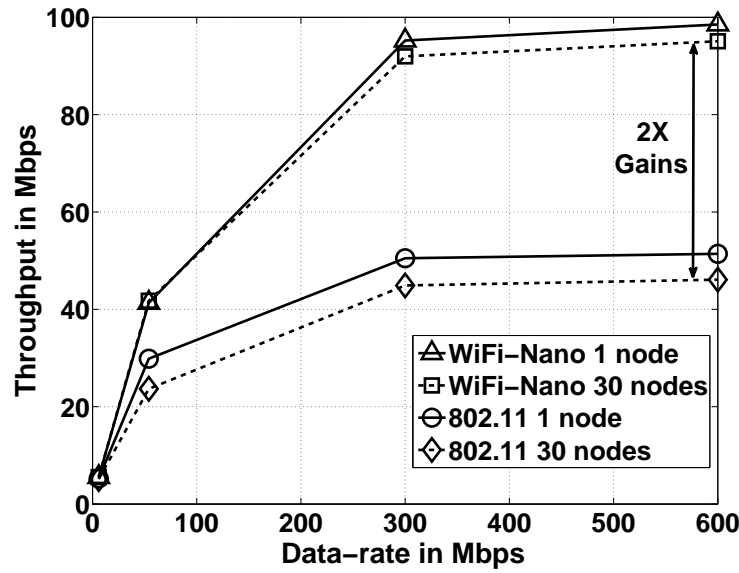


Figure 2.12 Throughput

Mbps, 54 Mbps, 300 Mbps, 600 Mbps, respectively. Next consider the case of 30 transmitters. The gap between WiFi and WiFi-Nano widens further since WiFi-Nano is able to mostly avoid collisions while the increased number of nodes reduces channel access overhead. On the other hand, the collision overhead in WiFi reduces aggregate throughput compared to the single transmitter case. Thus, at 300 Mbps and 600 Mbps, WiFi-Nano is able to achieve a throughput gain of 117% and 119% over WiFi, respectively.

Similar to Figure 2.2, I further analyze the performance of WiFi-Nano by decomposing the overhead into Preamble, ACK + SIFS, Channel Access, and Collisions. Figure 2.5.3 shows the results for 1 and 30 senders at data-rates of 54 Mbps, 300

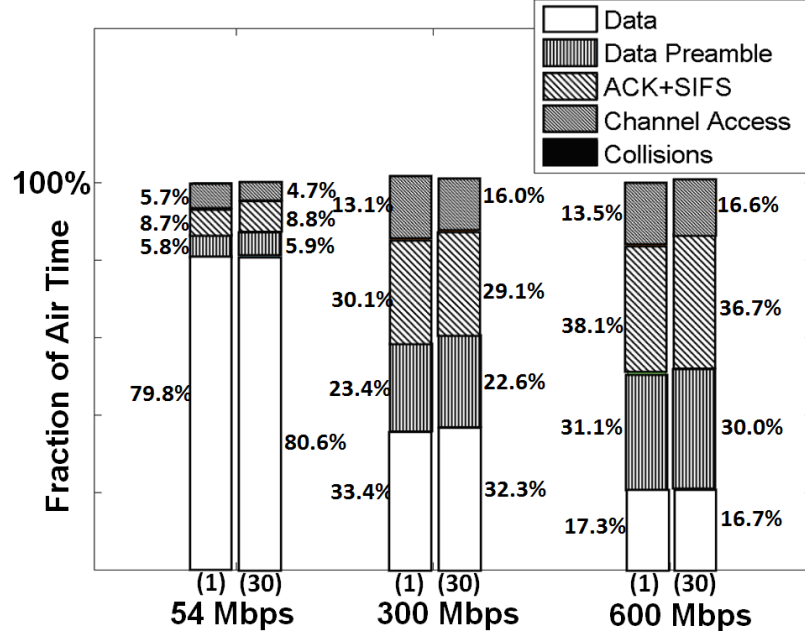


Figure 2.13 WiFi-Nano overheads

Mbps and 600 Mbps. I observe that the time spent in data collisions for the 30-node case is not visible, as collisions accounts for less than 1% of the time. Furthermore, the preamble and ACK overheads cannot be eliminated since they are essential for receiving data at these high rates (the SIFS overhead has been eliminated). Thus, the real overhead that remains in WiFi-Nano is only the channel access overhead which represents between 5.7-16.6% in the figure.

Fairness. Finally, I investigate the benefit of WiFi-Nano in terms of fairness. In 802.11, the backoff counters of two nodes may expire during the same slot, and thus cause a collision; depending on their relative SINR at the receiver, the receiver may be able to decode one of the packets (see capture effect [36]). This unfairly favors the

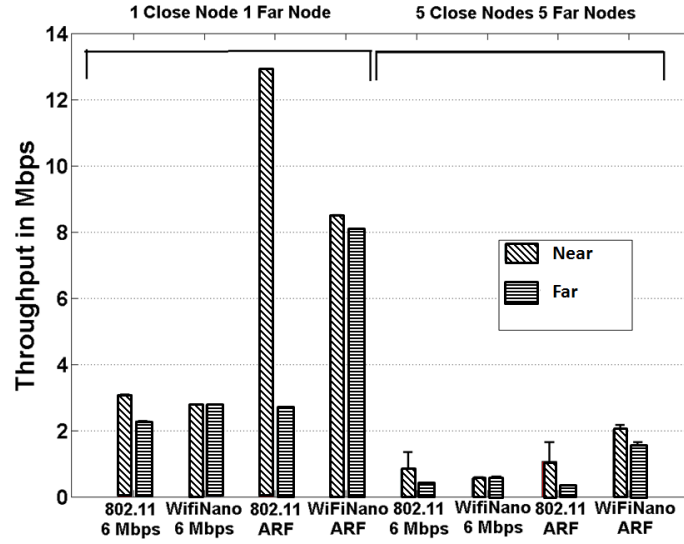


Figure 2.14 Fairness

nodes closer to an AP which benefit from a higher SINR. In WiFi-Nano, the preamble detection process terminates with a single node transmitting the data packet in the overwhelming majority of the cases; thus, the role of the capture effect is significantly reduced.

In this experiment, I used two kinds of nodes – near and far nodes. Near nodes are located 10 m from the AP while far nodes at 90 m from the AP. I consider both fixed rate transmissions at 6 Mbps as well as auto-rate fallback for 802.11a rates. I conducted two sets of experiments – first, with one node near and one far, and second with 5 nodes near and 5 far. Figure 2.5.3 shows the per-node average throughput for WiFi-Nano and 802.11 at different combinations of data-rate and number of nodes. Within each pair of bars, the left bar represents the throughput of closer node(s) while

the right bar depicts the throughput of the farther node(s); the error bars correspond to the maximum and minimum throughput of the nodes in each set.

Consider the case of one far and near node. In 802.11, the throughput of the closer node is 35% and 479% higher than the farther nodes when fixed rate and auto-rate fallback, respectively, are used while in the case of WiFi-Nano, the gap is less than 1% and 5%, respectively. Thus, WiFi-Nano is able to re-establish fairness without penalizing the total network throughput.

2.5.4 Frame aggregation

One way of reducing the overhead in WiFi is by transmitting larger packets so that the overhead is reduced to a small fraction of the transmission time. 802.11 standards have incorporated frame aggregation and block acknowledgements to allow frames of up to 64 kB to be transmitted at a time. Thus, data transmission time at 600 Mbps can be increased from $12\ \mu s$ to $853\ \mu s$, thereby reducing the overhead from 90+% to under 20%.

In Figure 2.5.4, I plot the efficiency (i.e., the ratio of the achieved throughput over the nominal data-rate used) of a single backlogged flow, as the aggregate frame size increases, for 300 and 600 Mbps data rates. Even for fully aggregated 64 kB frame sizes, WiFi-Nano is able to achieve 5-10% throughput gains over standard 802.11. However, 64KB average frame sizes are hard to achieve in practice due to

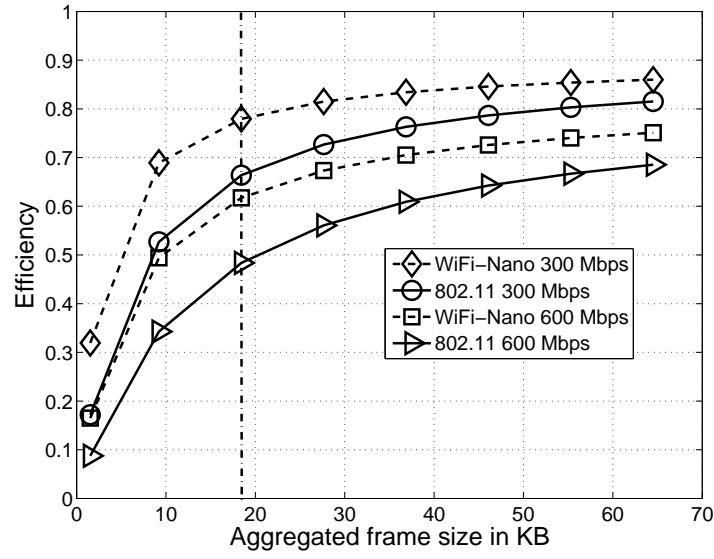


Figure 2.15 Frame Aggregation

the presence of small packets such as TCP ACKs or VoIP packets that cannot be aggregated. At typical average aggregation frame size of 18 KB [64], WiFi-Nano achieves 25% (resp., 17%) gain over 802.11 at 600 Mbps (resp., 300 Mbps).

Chapter 3

802.11ec

MAC control messages are essential: for example, ACKs convey correctly received data and RTS/CTS exchange can significantly mitigate hidden terminal collisions. However, even though the information conveyed in MAC control messages is small, their duration can be quite long, as in addition to the control information, they also need to include source/destination address, message type, etc., all of which are transmitted at base rate to improve the likelihood that they can be successfully decoded. For example, an ACK message is 14 bytes plus physical layer encapsulation, but contains only one bit of relevant information (that DATA was successfully received). Likewise, RTS/CTS is rarely used in practice precisely due to excessive overhead despite its important role in mitigating collisions.

In this paper, I design, implement, and evaluate 802.11ec (Encoded Control) as a control message free MAC. Instead of control messages, 11ec employs correlatable symbol sequences (CSS's), which together with their transmission timing, convey all control information, and change the fundamental design properties of the MAC. For example, 11ec replaces an 802.11 ACK *message* with a predefined ACK CSS that can be correlated instead of decoded, thereby vastly reducing its duration and dramatically improving its robustness by enabling its reception at low SINR.

Control information can be classified along two dimensions: first, as to whether or not the information in the message can be represented from a small dictionary or codebook. For example, a small dictionary can encode the three different control messages used in 802.11 for data exchange (RTS, CTS, and ACK). Likewise, while the space of all MAC addresses is large (seemingly precluding a small dictionary), each node communicates with only a limited number of addresses at a time. Thus, both MAC addresses and control message type can be encoded from a small dictionary. Second, control information can further be classified according to whether they are necessarily public or can be private. For example, for correctness of the protocol, all nodes must know that a CTS should cause them to defer, i.e., this message must be public; on the other hand, only a data sender need know that its data was correctly received, i.e., its ACK may be private.

802.11ec's key techniques are two fold: first, I use a dictionary of correlatable symbol sequences to convey control information that can be represented by a limited dictionary. For example, instead of CTS that contains physical layer preamble, frame control sequence, type field, frame check-sum, destination address, duration field (as well as it incurs a T_{SIFS} delay), I transmit a short (e.g., 127 symbols) CSS from a small dictionary to convey that it is a CTS. For an 802.11a physical layer, I show that this reduces the time to convey the control information by nearly an order of magnitude, from 60 μ s to 6.35 μ s. Second, I show that the information that cannot

be represented by a limited dictionary can be conveyed via CSS timing. For example, nodes overhearing the 802.11 CTS message need to defer for an amount of time as specified by the CTS duration field contained in the message. I show that 11ec nodes can instead simply defer until a channel-clear CSS is transmitted by the receiver (or until a timeout).

802.11ec's second technique is to distinguish between public and private information. Namely, 11ec only uses public CSS's for information that is required to be public, such as conveying channel reservation and channel clear. On the other hand, address fields need not be public, as the identity of the sender and receiver need not be known by other nodes. 11ec ensures that private control information, including addresses and ACKs, is not correlated by other nodes. This has the potential to thwart eavesdroppers not only from decoding data (as data can be encrypted), but even from knowing which nodes are communicating with each other; I show how all private control information, including addresses can only be correlated by the intended receiver.*

802.11ec enhances robustness in two ways. First, control information is more likely to be received in 11ec because control information is conveyed in short CSS's that are correlatable even at low SINR. For example, because 802.11ec replaces an ACK message with a CSS, 11ec ACKs are more robust and can be received even

*While development of a complete privacy protocol is beyond the scope of this work, 11ec provides important mechanisms to design such a protocol.

in the presence of transmitting interferers. Second, 802.11 is “fragile” to topological factors in that while 802.11 DCF without RTS/CTS yields high performance in fully connected wireless LANs [7], hidden terminals, asymmetric topologies, and general multi-hop topologies can yield severe throughput degradation and unfairness [10]. These latter topologies are becoming increasingly common because of device power asymmetries, e.g., between APs, laptops, and popular smart-phones, and of the wider coverage achievable with the adoption of sub-GHz frequencies, including TV white spaces [37, 55]. While use of RTS/CTS can significantly improve throughput in such challenged topologies, the additional overhead of RTS/CTS can sometimes overwhelm this improvement. Moreover, in fully connected topologies RTS/CTS degrades throughput due to its unnecessary overhead. In contrast, 11ec overcomes these limitations through robust and short-duration control signals, i.e., 11ec minimally penalizes station throughput thus allowing to enable channel reservation independently of the network topology. Consequently, 11ec stations have vastly increased opportunities to obtain channel access thereby dramatically improving the network’s fairness in throughput distribution.

I implement correlatable symbol sequences in a software defined radio, perform a large set of experiments and study issues that have not been experimentally investigated previously. I find a correlatable symbol sequence *length* that simultaneously: (i) provides sufficient physical-layer robustness, (ii) limits communication overhead, and

(iii) supports large networks. Specifically, I first investigate the trade-offs between sequence length and physical-layer robustness and show that even short sequences, e.g., 127-symbol or $6.35 \mu\text{s}$ long, can be detected at -6 dB SINR with only 5% false negatives. I demonstrate that my encoded sequences can be detected at an SINR 10 dB lower than 802.11 control messages. Second, I show that 127-symbol code lengths can support more than 50 co-located nodes, with minimal penalty on detection errors.

Finally, I implement 11ec in a measurement-driven emulator, whose inputs are channel measurements collected in a real deployment and real card performance parameters (e.g., BER and multiple supported modulations). I compare 11ec's performance to 802.11 with and without RTS/CTS. I examine a wide set of basic topologies that are at the origin of throughput losses and/or imbalances in 802.11-based networks in order to provide an insight in understanding the performance of larger networks. Our finding is that 11ec can dramatically reduce throughput imbalances by improving the Jain index [24] by up to 90%. Moreover, while such a fairness improvement can often decrease total utilization, 11ec increases channel utilization by more than 10% via the use of short encoded control that simultaneously decreases vulnerability intervals and control overhead. I also study a larger topology and show that 11ec can improve the throughput of an under-served flow by a factor of over 22x. Over all flows, I improve Jain index by up to 173% while also improving the channel utilization by up to 46%. Finally, I simulate larger 20-node topologies and show that 11ec

highly benefits the lowest throughput links, by at least doubling the throughput of more than 49% of them.

The remainder of the paper is organized as follows. Section 3.1 discusses coded control CSS's and the design of 802.11ec. Section 3.2 includes a thorough experimental evaluation of CSS's using a software defined radio platform. Section 3.3 investigates the benefits of 802.11ec in a measurement-driven emulator.

3.1 MAC Protocol Design

11ec collision avoidance realizes and improves on the collision avoidance mechanism of IEEE 802.11 DCF with RTS/CTS, reduces the overhead by nearly an order of magnitude, and practically eliminates collisions, even in hidden terminal topologies. Specifically, 11ec retains the four-way handshake suggested by 802.11, where control messages are replaced by very short correlatable symbol sequences (see Figure 3.1). It is important to note that: *(i)* the duration of each correlatable symbol sequence is nearly zero; *(ii)* the duration between the start of the reservation signal until the data transmission is negligible, hence practically invulnerable to collisions.

In this section, I define correlatable symbol sequences (CSS's) and explain how control messages can be turned into CSS's. Furthermore, I show that CSS's are a key element for the realization of 11ec efficiency and robustness. The second part of the section provides a detailed description of 11ec including protocol primitives,

and an analysis of hidden terminal vulnerability leading to novel collision reduction opportunities.

3.1.1 Coded Control Information versus Message Control Information

CSS's. Correlatable symbol sequences are predefined pseudo-noise binary codewords; namely, while codewords are deterministically generated, they retain the statistical properties of a sampled white noise. For this reason, the cross-correlation of any such sequence with a matching copy obtains spike values, while it appears random to a listener without prior knowledge of the codeword. An example of a CSS is the 802.11 preamble used for packet detection, symbol synchronization, and radio parameter tuning.*

The CSS detection process via cross-correlation enjoys three key advantages over data decoding. First, cross-correlation obtains a large processing gain even for small codewords (e.g., the 802.11 preamble used for detection is 64 symbols), which permits reliable detection even at low SINR. Second, differently from decoding, detection is highly robust to imperfect radio parameter tuning and thus a codeword does not need to be preceded by a preamble. For these reasons, a CSS can be short; for example, in my implementation, 11ec utilizes 127-symbol codewords that can be transmitted in $6.35 \mu\text{s}$. Third, detection is almost instantaneous as no decoding is needed. For

*A detailed discussion of signal correlation can be found in literature [21, 34, 60]. A short introduction is in Appendix 3.A.

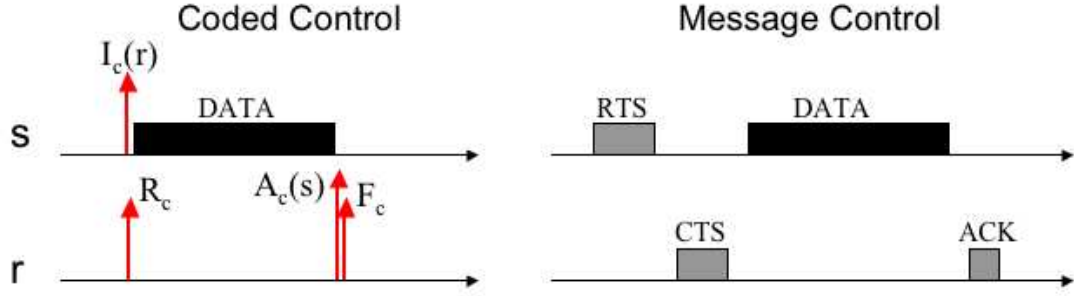


Figure 3.1 Timeline of a packet exchange with Coded Control versus Message Control. example, 802.11 inter-control message time is at least $T_{SIFS} = 16 \mu s$, including about $14 \mu s$ of data processing, while in 11ec no control message processing is required; hence, 11ec reduces substantially the short inter-CSS time.

Encoded Control. While consuming a significant amount of airtime, 802.11 control messages usually convey little information. For example, an ACK occupies the medium for up to $60 \mu s$, i.e., an airtime sufficient to transmit 3240 bits at 54 Mbps, while it contains a single bit of relevant information. 11ec replaces control messages with CSS's, which permit to shorten the transmission duration of nearly an order of magnitude to $6.35 \mu s$, while retaining the information content.

According to the 802.11 standard [1], RTS, CTS, and ACK control messages may include up to fmy information fields: destination address, sender address, duration, and frame control (a fifth field is the frame check-sum that protects the other four). In particular, the frame control field is a 2-byte long sequence of bits representing specific control parameters. The values of most control bits are fixed for control

messages; only frame subtype (4 bits) and station power management flag (1 bit) can assume different values. However, the latter does not convey novel information when used in control messages.

In order to represent the information content of the control messages as described above, 11ec considers the size of the dictionary needed to represent such information. Information that can be expressed by a small dictionary is conveyed using CSS's, while information that needs large dictionaries is conveyed with timing codes. First, in 802.11 data exchange, the type field that distinguishes the control messages may assume only three values, i.e., RTS, CTS, and ACK; 11ec conveys the type by associating each message with distinct CSS's. Second, 802.11 control messages include addresses (sender and/or receiver). Since the number of nodes a station communicates with at a time is generally small, i.e., can be represented by a small dictionary, 11ec integrates the addresses in CSS's, i.e., a single CSS may represent the combination of a control type and a specific address. For example, in 802.11 the RTS includes the address of the intended receiver; accordingly, in 11ec, RTS addressed to different receivers are represented by distinct CSS's. Third, some control messages include a duration field that cannot be represented by a small dictionary. For this information, 11ec utilizes a combination of time codes and new control types. For example, 11ec nodes reserve the channel for the duration of a data reception, by transmitting two CSS's corresponding to channel reservation (immediately before the reception) and

release (immediately after). The potential interferers do not access the channel during the interval between the two CSS's (or before a timeout expires), i.e., effectively implement a form of virtual carrier sensing.*

Public CSS's versus Private CSS's. A second dimension of control messages, and of the corresponding CSS's, is whether they can be private, i.e., carry information relevant only to a specific destination (e.g., acknowledgements), or are necessarily public, i.e., meant to be heard by all neighbors of a node (e.g., channel reservation/release). Accordingly, in the case of a private CSS, only the intended receiver possesses a copy of the correlatable symbol sequence, and thus can correctly detect it; conversely, all nodes possess copies of the public CSS's and can detect them. For example, only the data sender needs to cross-correlate its private acknowledgement CSS, while all nodes must cross-correlate public channel reservation/release CSS's.

Control during Data. Because correlatable symbol sequences can be detected even at sub-noise SINR (e.g., 11ec 127-symbol CSS's can be detected at -6 dB with high reliability), 11ec nodes attempt detection even while receiving data. This technique provides a signaling mechanism effective even in cases in which the receiver is subject to long periods of noisy channel, or undesired data overhearing. In 802.11, if a node receives an RTS while also overhearing data, the node cannot decode the

*Another alternative is to discretize the duration and associate different CSS's to each discrete value.

RTS and therefore cannot respond. In contrast, 11ec uniquely enables a node to receive a CSS signaling that another node is requesting to communicate. Therefore, the receiver can send a Request for RTS (RRTS) CSS to reserve the medium when it becomes free and initiate a data exchange [5]. Likewise, because ACKs are also encoded, they can be correctly received even if an interfering terminal is simultaneously transmitting data.

3.1.2 11ec Channel Reservation Primitives

Wireless MAC protocols perform collision avoidance by silencing the medium in the vicinity of a transmitting link via channel reservation. Channel reservation fundamentally hinges on three key mechanisms: *(i) initiation*, performed by the node that initiates the exchange to request the cooperation of the other endpoint to reserve the channel; *(ii) reservation*, performed to inform nodes potentially hindering the exchange; and *(iii) deferral*, performed by the surrounding terminals in order to avoid disturbing ongoing transmissions. 802.11 implements the three mechanisms via *(i)* RTS; *(ii)* CTS and data packet - the latter realizes channel reservation in the vicinity of the sender; and *(iii)* NAV and carrier sensing. When RTS/CTS is disabled, during the data transmission the medium is reserved exclusively in the vicinity of the sender. In the following, I show how 11ec implements these three mechanisms via CSS's and timing codes.

A key concept of 11ec channel reservation is very short channel reservation negotiation for near immunity to interruptions, e.g., collisions and capturing by other nodes. Specifically, 11ec channel reservation is based on three basic primitives (the subscript c indicates CSS's).

Initiation: $I_c(r)$. In 11ec, a sender wishing to start a data exchange performs virtual, and optionally physical, carrier sensing. If the medium is free, the sender waits for a backoff interval similar to 802.11 and then transmits a sender side channel request primitive, in short $I_c(r)$, to request the receiver r to reserve the channel. $I_c(r)$ need only be detected by r and not necessarily by the neighboring nodes; thus, I implement it as a private CSS. In order to convey the identity of the receiver r (as 11ec does not transmit the traditional MAC address), 11ec implements $I_c(r)$ via several CSS's, and associates a distinct CSS with each receiver; i.e., when a sender needs to contact a receiver, it uses the receiver's $I_c(r)$. Nodes in the vicinity of the sender do not detect the initiation $I_c(r)$.

Reservation: R_c . A node r receiving an $I_c(r)$ checks if other nodes are communicating in its vicinity and may hinder its reception. If that is not the case, r immediately transmits a channel reservation primitive R_c to notify potential interferers. In order to realize the reservation, R_c should be detected by all nodes in the vicinity of the receiver r and it is therefore transmitted via a public CSS.

In addition to channel reservation that forces neighbors to defer, R_c implicitly

communicates to the transmitter that the channel is available and that data can be transmitted. Instead of providing a distinct CSS to convey the sender address, as in the previous case of the $I_c(r)$, I employ a simple temporal code: Since a receiver r transmits R_c immediately after $I_c(r)$, the sender, in contrast to the other neighboring nodes, interprets the reception of a R_c as an authorization to begin a data transmission.

Deferral: $R_c \rightarrow F_c$. 11ec implements the deferral and conveys its duration via a combination of CSS's and a simple time code. Specifically, after data reception and acknowledgement, the receiver explicitly releases the channel with a channel free primitive F_c . Thus, nodes receiving R_c need to wait to receive an F_c (or wait a pre-defined timeout) before accessing the channel; practically, this procedure represents a form of virtual carrier sensing. Because all neighboring nodes need to receive F_c , 11ec implements F_c as a public CSS.

11ec further increases the robustness of R_c/F_c messages by pairing them. Our technique is based on associating a small number of CSS pairs to distinct R_c^i/F_c^i pairs; a receiver randomly picks and transmits any such pair to reserve and free the channel. This feature is particularly useful for a node located in the neighborhood of several receivers that may be active simultaneously, i.e., the receivers may send R_c and F_c that denote overlapping intervals. In that case, the node can still correctly determine the state of the medium in each moment, by associating each overheard

reservation R_c^i to its corresponding F_c^i .

Finally, I define an acknowledgement primitive, $A_c(s)$ (see Figure 3.1), and I implement it as a private CSS associated to each sender s . Few recently proposed packet forwarding schemes leverage the ACK exchange for additional purposes other than data acknowledgement, e.g., network coding [15] and routing [16]. In such cases, 11ec can revert to 802.11 ACKs with small overhead penalty (the major gain of 11ec both in throughput and robustness derives from the novel channel reservation scheme). Note that CSMA/CN [60] uses signatures for acknowledgement, which are similar to my CSS's.

Primitive Extensions. As briefly mentioned above, 11ec can support a signaling mechanism to alleviate starvation similar to RRTS [5] via control during data, and can highly reduce the occurrence of exposed terminals via robust CSS acknowledgement. For reason of space, these cases are not covered in this paper.

3.1.3 Discussion

In this section, I discuss three issues involving the protocol primitives presented above: private CSS association, power consumption, and semantic divergence from 802.11.

Private CSS Association. The initiation and acknowledgement primitives $I_c(r)$ and $A_c(s)$ require association of unique CSS's to each network node. * Specifically,

*This also requires that different networks operating in the same locality use non-overlapping sets of private CSS's, while they may use identical channel reservation/free CSS's.

two nodes may reuse identical CSS's if each of the two is out of the range of the other node's potential CSS transmitters. This requirement prevents that any of the two nodes wrongly assumes that a CSS is targeted to itself, when in fact the CSS is not.

While a detailed investigation of private CSS association is beyond the scope of this paper, I suggest two simple mechanisms that can be used. The first leverages the solution already existing in the 802.11 standard, in which the AP assigns an identifier (AID) upon station association [1]; AIDs can be easily mapped to CSS's. In order to initiate the association, the stations may use a reserved CSS. The second mechanism utilizes multiple hash functions based on the station MAC address. In case of assignment conflicts, the stations can switch the hash function utilized. Note that 11ec nodes can easily detect assignment conflicts. In fact, in the case of $I_c(r)$ conflict, the subsequent data header will indicate the actual identity of the destination; in the case of $A_c(s)$ conflict, a node will receive the CSS without any preceding data transmission.

Power Consumption. 11ec enables key efficiency gains that allow considerable power savings, e.g., by reducing the overhead. In order to achieve this benefit, each node may need to activate several correlators simultaneously. In fact, while in idle state, any 11ec node x needs to be able to detect at least three incoming CSS's, namely the private initiation $I_c(x)$, and the public pair channel reservation R_c /channel free F_c (the acknowledgement $A_c(x)$ needs only be detected immediately

after a packet transmission). While this may raise concerns over the node power consumption, the very loose upper bounds available in literature are encouraging. According to [48, 65, 73], the whole baseband processing of a radio device (which includes a single correlator and a full receiving chain) may consume about 27% of the total reception power of a radio. Even attributing all such consumption to the correlator, the total power increase due to the use of three correlators for a node in idle state would be approximately 50% (of course the final figures depend on implementation technique and CSS's sequences in use). Finally, 11ec may prevent device power-off opportunities [52], due to the requirement that nodes continuously monitor the state of the channel.

Semantic divergences with respect to 802.11. 11ec CSS's convey an information content that, while close to 802.11's control messages, differs from it in several subtle ways. In the following, I examine the most relevant differences and their consequences. While some of these issues may seem to give rise to critical sequences of operations, it should be noted that CSS's robustness to interference and short duration tend to dramatically reduce their incidence, as shown by the results in Section 3.3. Furthermore, most of the critical sequences could be avoided using a larger set of CSS's.

(i) $I_c(r)$ does not include the identity of the sender; thus, r cannot indicate the identity of the designated transmitter in R_c . This differs from 802.11, where RTS

and CTS both identify the sending endpoint of the data exchange. In 11ec, if two senders initiate a communication (not necessarily towards the same receiver) at short time distance, both of them may interpret the subsequent R_c as an authorization to transmit. The next section shows a technique to address this issue.

(ii) $I_c(r)$ may not be detected by the neighbors of the transmitter, thus not reserving the exclusive use of the medium, in particular for the reception of the R_c . In contrast, 802.11's RTS permits neighbors to NAV. In 11ec, two neighboring senders may initiate a communication with different receivers at short time distance, i.e., the latter may send an $I_c(r')$ while the first is waiting for R_c from r . This situation may potentially improve the spatial reuse in case the two senders are exposed terminals; otherwise, it may lead to increased interference and transmission failures.

(iii) R_c does not indicate the duration of the subsequent data exchange. In contrast, as mentioned above, 802.11's CTS explicitly contains such duration. In 11ec nodes may receive an R_c , but miss the corresponding F_c ; to contrast this issue, 11ec nodes adopt the timeout strategy described above.

3.1.4 Contending Flows and Vulnerability Interval

The shortness and robustness of correlatable symbol sequences dramatically reduces vulnerability to collisions. The vulnerability interval of a packet exchange is twice the time delay from the beginning of a transmission until all potential interferers are

prevented from corrupting the exchange, i.e., it includes the whole interval, before and after the beginning of the intended exchange, during which interfering transmissions may start and corrupt the exchange. In the case of hidden terminals in **802.11** with RTS/CTS, the vulnerability interval is twice the delay from the moment a node sends an RTS to the detection of CTS by the hidden terminals. Considering the case of 802.11a/g and 6 Mbps control packets, the total duration of the vulnerability interval can be computed as follows. First, node s transmits the RTS, for a total duration of $52 \mu\text{s}$ including preambles; second, the RTS propagates to the receiver for up to $1 \mu\text{s}$; third, the receiving node r waits $T_{SIFS} = 16 \mu\text{s}$ before sending the CTS, due to practical communication issues such as RTS decoding; fourth, the CTS propagates to hidden terminals for up to $1 \mu\text{s}$; fifth, the hidden terminals need up to $4 \mu\text{s}$ to detect the packet; last, additional $2 \mu\text{s}$ account for potential radio turn-around (all temporal indications are taken from section 17 of the 2007 version of the standard, for 20 MHz bandwidth [1]). The total amounts to $152 \mu\text{s}$, i.e., twice $76 \mu\text{s}$. In 802.11 without RTS/CTS, the vulnerability interval can be considerably larger, spanning twice the data transmission duration and as large as 4 ms.

802.11ec's CSS's shorten the vulnerability interval and practically reduce the set of potential hidden terminals. The vulnerability interval of 11ec can be computed as follows (see Figure 3.2, where the intervals below are denoted by numerical time indications). First, node s transmits $I_c(r)$ for $6.35 \mu\text{s}$; second, the receiving node

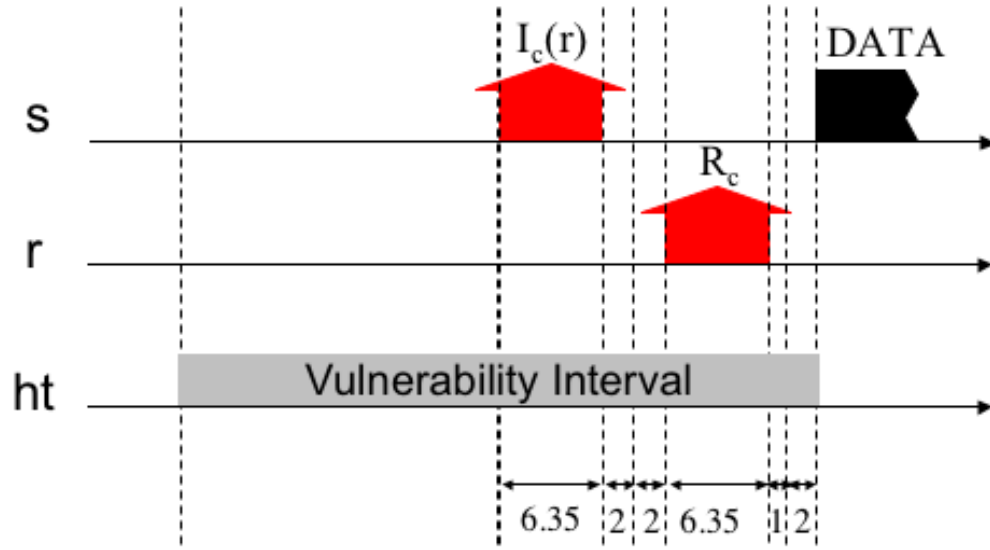


Figure 3.2 Timeline of the vulnerability interval of 802.11ec (time indications are in μs).

r waits for up to $2 \mu s$, in order to detect potential overlapping hidden terminal transmissions as explained below (this interval is a design choice and includes the propagation delay from the transmitter); third, *r* needs a turn-around time to prepare its radio for transmission, i.e., $2 \mu s$ (according to 802.11 standard); fourth, *r* transmits R_c for $6.35 \mu s$; finally, the hidden terminals need a propagation delay of up to $1 \mu s$ to receive the R_c and an additional $2 \mu s$ to account for potential radio turn-around. 11ec vulnerable interval is twice $19.7 \mu s$, i.e., $39.4 \mu s$ or about 25.9% of the vulnerability interval of 802.11.

CSS's also nearly eliminate the collisions of nodes starting in the same slot when

the SINR allows it.* In fact, the receiver r can detect simultaneous and overlapping transmissions of multiple $I_c(r)$ because of the CSS processing gain, and request re-transmission. Specifically, r waits for a round-trip propagation time in order to detect potentially overlapping transmissions and, in that case, sends a negative acknowledgement which prompts the contending nodes to undergo a quick backoff repetition. However, in order to take advantage of this technique, I need to enlarge the slot size to encompass the half vulnerability duration, i.e., $20 \mu s$ (note that this is sufficient, as the F_c of the receiver synchronizes all of its transmitters). Also in cases of no hidden terminals, this choice induces only minor throughput penalties at high data rates, as I show in Section 3.3.2.

Co-existence with legacy 802.11. For backward compatibility with 802.11, 11ec exactly follows the standard [1] except for the techniques described in this section. For example, a key element for co-existence is the arbitration of the medium, which leverages carrier sensing based on the correlation of the data preambles and backoff mechanism. Accordingly, 11ec uses the same data preamble format as 802.11, and sets the contention window size following the same binary exponential backoff scheme.

A more complete discussion of co-existence is beyond the scope of this paper.

*Nodes may use power adaptation techniques exclusively to transmit control CSS's while transmitting data at full power, i.e., without modulation rate adaptation requirement or throughput penalty. While this may improve the performance of 11ec, I defer its investigation to future work.

3.2 Experimental Evaluation of CSS

In this section I present an experimental evaluation of correlatable symbol sequences using software defined radios. Specifically, my evaluation covers the following issues, *none of which has been previously experimentally studied in the literature.*

(i) I explore the trade-off between length of the sequences, i.e., overhead, and processing gain, i.e., robustness. **Our finding is that 127-symbol sequences provide a good trade-off between overhead (6.35 μ s) and robustness (5% false negatives at -6 dB SINR).**

(ii) I contrast the performance of CSS detection with control message decoding. **Our finding is that 127-symbol CSS's can be reliably detected at about 10 dB lower SINR than 802.11 6 Mbps OFDM control packets.**

(iii) I determine the codebook size that 11ec can support, i.e., the number of distinct CSS's that can be practically used, by studying the cross-correlation between different CSS's and its effect on the probability of false positives. **Our finding is that the design of 127-symbol CSS's via Gold codes can support more than 50 co-located nodes (a total of 127 CSS's), without any penalty on false positives.**

3.2.1 Experimental Setup

3.2.1.1 Tools

WARP and WARPLab. Our reference software defined radio is the WARP platform [3]. WARP is an FPGA-based platform, including custom designed radios based on the MAX2829 chipset. WARPLab is a programming environment that permits to drive WARP from a host computer. Relevantly to my experiments, WARPLab supports the execution of micro experiments, each one of approximately $400\ \mu\text{s}$ duration (2^{14} samples at the 40 MHz frequency of DAC/ADC), and access analog sample send/receive buffers and RSSI recordings collected during each experiment. RSSI is measured by the MAX2829 circuit, and digitized by a dedicated 10-bit ADC.

Azimuth Channel Emulator. In order to perform experiments under controllable and repeatable conditions, I used an Azimuth ACE MX channel emulator.* The channel emulator permits creation of different network topologies, by tuning the attenuation along each path independently and predictably.

3.2.1.2 Implementation

I implement CSS transmission/detection and OFDM packet transmission/decoding on WARPLab. Specifically, CSS's are BPSK sequences filtered, upsampled, and transmitted via standard wideband methods. This solution enjoys a practical advantage

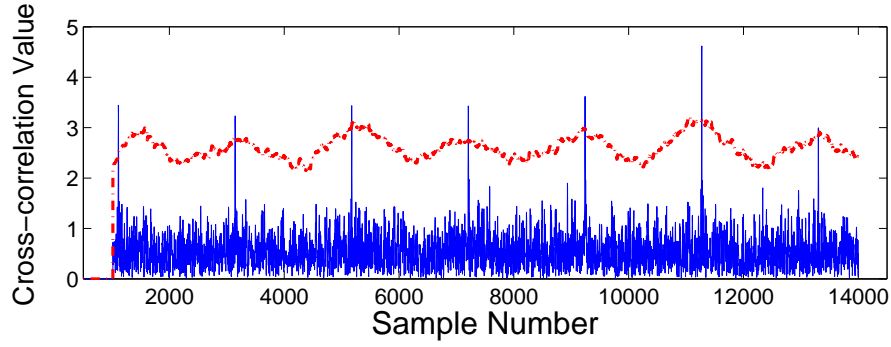
*Azimuth Systems Inc., <http://www.azimuthsystems.com/>

over alternative solutions, e.g., OFDM-modulated BPSK sequences, due to the lower peak to average power ratio [68]. Finally, in order to reproduce 802.11 as closely as possible, I implement all types of OFDM 802.11a/g modulation and convolutional code pairs in WARPLab.

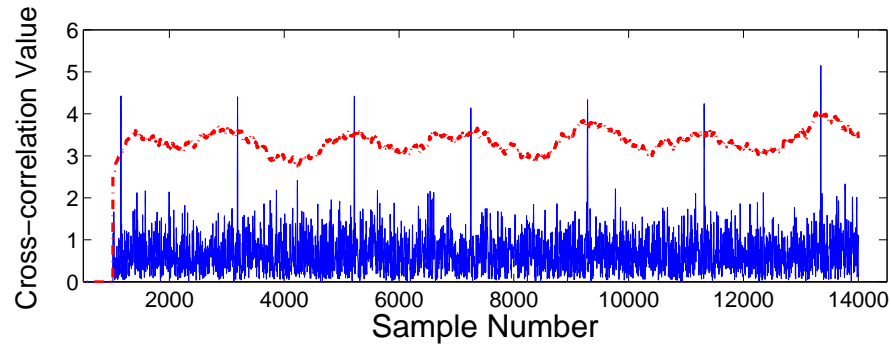
3.2.2 Channel Emulator Validation

The results in this section are obtained using a channel emulator. In order to validate the emulator setting, my methodology includes a preliminary validation contrasting results of a cross-correlation experiment performed over the air, with an identical experiment conducted with the emulator. Specifically, my experiment consists of exchanging CSS's between two WARP nodes a and b , under the interference generated by random OFDM transmissions of a third interfering node c . In the first part of the experiment, I deploy the three nodes inside an office building. In an over the air setting, it is difficult to control SINR given interference from 802.11 networks operating in the building. For this reason, I perform the experiment late at night and measure the SINR on links $a-b$ and $c-b$ a few seconds before and after the experiment. Then, I repeat the experiment under controllable and repeatable conditions using the channel emulator, where I can control the SINR with high accuracy. I repeat both experiments several times and for different SINR, and show a representative sample result.

In both experiments, node a transmits 7 repetitions of a 127-symbol CSS. For each newly acquired sample (i.e., at 40 MHz frequency of the WARP platform ADC), node b computes the signal correlation with a local copy of the transmitted CSS. Figures 3.3(a) and 3.3(b) show a representative outcome of a realization in which the SINR on link $a - b$ carrying the CSS is -6 dB. The x-axis is the temporal progression of collected samples, while the y-axis is the correlated value. The thick crossing line in the plots represents a possible choice of the detection threshold; such a threshold is strategic in determining the robustness of CSS's, by balancing false positives and false negatives. In the experiments I conduct in this section, the threshold is chosen in order to obtain a false positive probability of 10^{-8} . While I defer more details on how to tune the threshold to Section 3.2.6, here I observe that because of the threshold design the correlation value on the y-axis is normalized according to the magnitude of correlated I/Q samples. In the figures, correlation spikes are clearly identifiable in coincidence with the reception of each single CSS, as all and only marks exceeding the threshold. Thus, the detection of 127-symbol CSS's is possible at -6 dB with few errors. *By comparing the two plots, I conclude that controllable emulator experiments and over the air experiments provide similar results for identical SINR values.* However, because of the difficulty to constantly control the SINR in over the air settings, I perform the remaining experiments in this section using the channel emulator, thereby also ensuring their repeatability.



(a) Over the air



(b) Channel emulation

Figure 3.3 Example of 127-symbol CSS correlation at -6 dB.

3.2.3 CSS Length versus Robustness Trade-off

The first issue is the trade-off between the CSS length L and robustness under different SINR; I quantify robustness in terms of the probability of false positives and false negatives. The outcome of this assessment is important, as robustness to SINR is one of the two key elements in the choice of CSS length (the other element guiding this choice is the number of CSS's, discussed in Section 3.2.5). In this subsection, I determine a length that can tolerate significant interference without high communi-

cation overhead. In this experiment, I deploy the three node topology above and use the channel emulator to vary the SINR on link $a - b$. Specifically, the link between node a transmitting the CSS and the receiving node b is maintained fixed to -82 dBm, while the attenuation on the interfering link $c - b$ is set in order to obtain the desired SINR. I iterate the experiment for different combinations of SINR and CSS lengths. Each experiment consists in the detection of at least 100 CSS's of lengths L ranging from 63 to 511 symbols. I vary the SINR between 0 dB and -10 dB.

Figure 3.4 shows the probability of false negatives as a function of SINR and CSS length. Specifically, the x-axis denotes the SINR on link $a - b$, while the y-axis denotes the probability of false negatives. The different curves correspond to different CSS lengths. The figure shows that longer CSS's are more robust due to the processing gain, e.g., at -8 dB, CSS's of length 63 can be detected only 4% of the time (96% of false negatives in the figure), while CSS's of lengths 127, 255, 511 can be detected approximately 30%, 99%, and 100% of the time respectively. However, increasing the CSS length involves an overhead penalty; in fact, while a 63-symbol CSS can be delivered in about $3.15 \mu s$, 127, 255, 511-symbol CSS's require 6.35, 12.75, 25.55 μs respectively. With regard to the probability of false positives, I never obtained more than a single occurrence (out of hundreds of thousands of tests performed) for all the experiments related to a fixed SINR and length combination. Finally, it is relevant to notice that my results show only minor degradation with

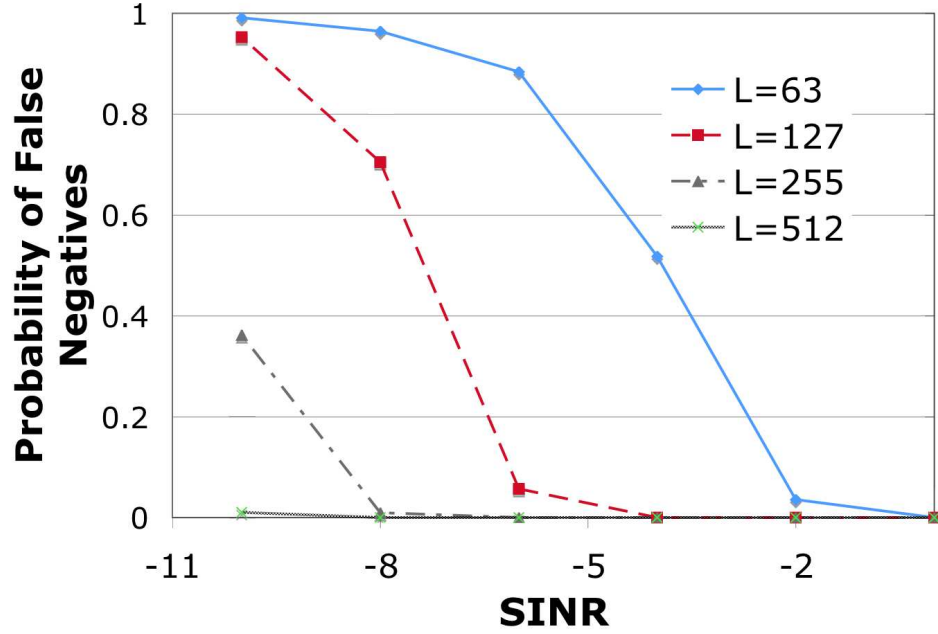


Figure 3.4 Robustness vs length tradeoff for different CSS lengths.

respect to theoretical performance in AWGN channel. For instance, considering the probability of false positives and false negatives at -8 dB, the length of sequences with similar performance in AWGN would be 47, 81, 198 for the cases of 63, 127, 255 actual lengths. *I conclude that 127-symbol CSS's provide a good compromise between overhead (6.35 μ s), and resilience as they can be detected at -6 dB with 5.7% false negatives and no false positives.*

3.2.4 CSS Detection versus Control Message Decoding

Our second experiment aims to show that control CSS's are more robust to noise than 802.11 control messages, i.e., that CSS's can be reliably detected at considerably

lower SINR than control messages. The metrics I use are false positives for the case of CSS detection, and packet error rate for control message decoding. I consider 127-symbol CSS's vs. 160-bit messages transmitted via BPSK modulation, with 1/2 rate convolutional coding, corresponding to an RTS packet transmitted at base rate of 6 Mbps in 802.11 OFDM. In order to create the interference scenarios, I follow a methodology identical to the previous experiment. For the case of BPSK modulation, my experiment directly measures the BER out of at least 100000 bit transmissions. Then, I convert the obtained value to packet error rate, by considering a random and independent distribution of the bit errors among the packets (i.e., $1 - (1 - BER)^{PL}$, where PL is the packet length in bits). Note that the adoption of burst error models, such as Gilbert-Elliot [18], with expected burst length of 6 bits [71] may vary the results by about 1 to 1.5 dB.

Figure 3.5 shows two curves corresponding to CSS detection and control messages decoding. The x-axis denotes the SINR, while the y-axis denotes the *probability of missing control*, i.e., the probability of false negatives (resp. of packet decoding error) for CSS's (resp. for control messages). The plot shows that control CSS's are substantially more robust than control messages, since their probability of false negatives is much less than the error probability of control packets for any SINR. Furthermore, similar *probabilities of missing control* are obtained for the two control mechanisms, for SINR values separated by about 10 dB. For example, CSS's obtain

	802.11b	802.11g	802.11a	802.11ec
Overhead	980	676	172	26.05

Table 3.1 Control Information Duration Overhead (times are in μs).

probability of false detection of 5.7% at -6 dB, while control messages achieve 24% packet error rate at +4 dB, and 0.2% at +4.5 dB. *I conclude that due to the improved robustness of CSS detection with respect to packet decoding, control CSS's are about 10 dB more resilient to noise than 802.11 control messages.*

Overhead Analysis. In Table 3.1 we contrast the overhead duration of control information in different 802.11 versions to 11ec. The times indicated encompass the transmissions of RTS/CTS/ACK, and the two SIFS intervals between RTS and CTS, and CTS and data packet for 802.11; the analogous intervals are considered for 11ec. The table shows that 11ec can reduce the transmission times by as much as 84.8% with respect to 802.11a (or 802.11g in the absence of 802.11b terminals), and of 96.1% (97.3%) with respect to 802.11g (802.11b). The observed gains are mainly due to (i) CSS simplified physical layer operations, which permit to remove SIFS and packet preambles, and to (ii) the reduced amount of information that CSS's carry, i.e., 7 bits for 127-symbol CSS. In order to separate the contributions of the two components, I finally assume a hypothetical hybrid solution in which 7 bits of control information are transmitted according to 802.11 physical layer operations. In that case, the control

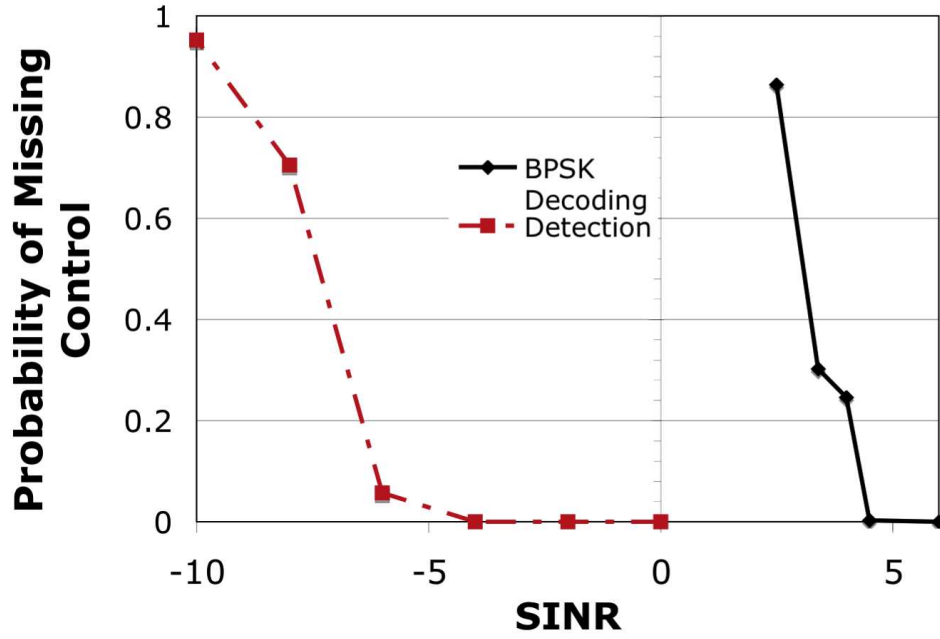


Figure 3.5 Probability of missing CSS detection vs missing message decoding.

overhead would amount to 104/608/617 μs for 802.11a/g/b resp., for a 11ec gain of 64.9%/95.7%/95.8%.

3.2.5 11ec Codebook Size

In 11ec, nodes use multiple CSS's and need to be able to reliably detect and discern all of them. In this experiment, I investigate whether cross-correlation between CSS's affects detection accuracy, and I explore the number of distinct CSS's that can be practically used. Specifically, I assess the probability of falsely detecting CSS A when CSS B is transmitted instead. For a given CSS length, a trade-off exists between the number of CSS's that 11ec uses, and the magnitude of the cross-correlation between

any CSS pair, which in turn influences the probability of false positives. For this error probability to be small, I use well known sparse binary sequences, with optimal cross-correlation properties. Instances of such sequences have been studied for lengths corresponding to powers of 2, e.g., in cellular communications [59]. Different families of sequences provide larger (resp. smaller) sets of codewords, with larger (resp. smaller) cross-correlation between any codeword pair. Our design implements Gold codes, which provide 127 CSS's for my 127-symbol length, with a theoretical cross-correlation on the order of 12%. The choice of Gold codes permits us to support more than 50 co-located nodes by assigning distinct CSS's pairs to each node representing $I_c(r)$ and $A_c(s)$, while saving several CSS's for F_c^i/R_c^i pairs. In case a larger number of nodes needs to be supported, 11ec can switch to 255-symbol Kasami large codes for example, which allow more than 2000 nodes with 4011 CSS's.

To verify my choice, I emulated a situation in which a CSS A is sent, and 10 nodes try to detect CSS's different from A within the same samples. I repeated the experiment for 100 detection attempts, for 127-symbol Gold codes and SINR from 0 dB to -10 dB. The goal of this experiment is to assess the probability that the other nodes obtain false positives of their own CSS when the signature A is sent. The number of false negatives is immaterial in this experiment. For each SINR experiment, I obtained at most one false positive more than the case of a single CSS detection. Figure 3.6 shows an example outcome for -6 dB SINR (where the axes have the same

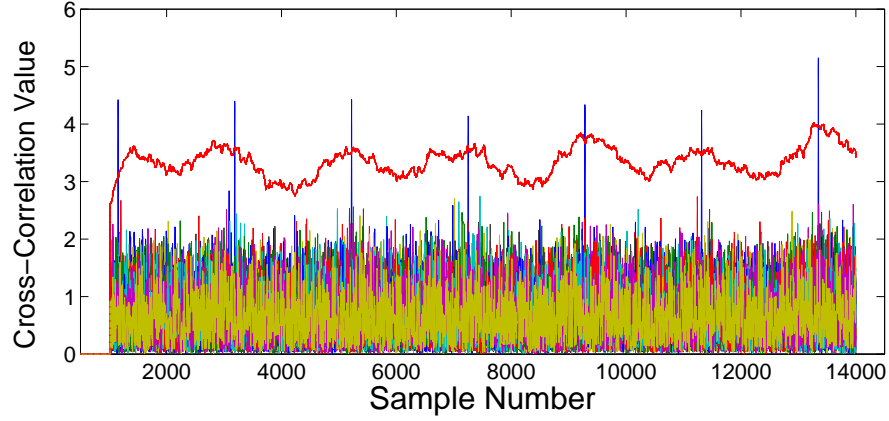


Figure 3.6 Low cross-correlation of CSS's different from the one transmitted.

meaning as in the experiment in Figure 3.3). The overlapping plots show the cross-correlation values obtained by 11 different nodes (including the one that expects to detect the transmitted CSS). While the number of spikes is unchanged, the noise looks visually denser due to overlapping plots. *I conclude that by using Gold codes, 11ec can support more than 50 co-located nodes without significant incidence of false positives.*

3.2.6 Discussion on Signal Correlation

Practical Detection Threshold Selection. The choice of the detection threshold is strategic in balancing the trade-off between false positives and false negatives. For example, as mentioned above, in Figures 3.3(a) and 3.3(b) the threshold is denoted by the thick crossing line; corresponding to the chosen threshold, the figures show 0 false positives and 0 false negatives. In general, a higher value of the threshold decreases

the probability of false positives at the expense of a large probability of false negatives; viceversa, a lower threshold increases the occurrence of false positives. The theory of correlation in Gaussian noise provides an optimal threshold for a target detection SINR [34]. Specifically,

$$T = \sqrt{\frac{L \cdot \mathcal{E} \cdot \mathcal{N}}{2}} * Q^{-1}(P_{FA}) \quad (3.1)$$

where Q is the tail probability of the standard normal function. The formula shows that the optimal T depends on noise power \mathcal{N} , CSS power \mathcal{E} , CSS length L , and on the target probability of false positives P_{FA} that I fix to 10^{-8} . I remark that: *(i)* Generally, the power of the noise (which may be due to interfering transmissions) varies in time, thus the detection threshold should also change; *(ii)* Practically, it is difficult to estimate in advance the power of the noise and the power of the signal. In order to address the latter concern, I establish a lower bound on the SINR of the signal that I aim to detect (in my experiments -6 dB), and I tune T correspondingly (i.e., $T = \sqrt{\frac{L \cdot \text{SINR} \cdot \mathcal{N}}{2}} * Q^{-1}(P_{FA})$). Unfortunately, this solution is not sufficient because of the difficulty to estimate \mathcal{N} . In fact, the receiver can only measure the total power of the incoming signal, which may or may not contain the target CSS. Thus, I conservatively choose to replace \mathcal{N} with the total signal power received, as if the incoming signal did not contain the CSS; practically, when the CSS is actually present, this choice has the effect of tuning T to a higher value than desired, i.e., it increases the occurrence of false negatives. Figures 3.3 and 3.6 show that the value

of the threshold increases when the signal is present, and decreases otherwise.

Wireless Communications Issues. It is important to note that two issues may affect the performance of correlation, both due to the fact that the transmitter and receiver radio generate independent clocks [21, 60]. First, the clock phases at the transmitter and receiver are in general not aligned; this produces a *phase offset* between the two radios, which causes a fixed rotation of the received symbols of an angle γ . In order to compensate for this effect, I compute the magnitude of the correlation, with a theoretical penalty on the processing gain of about 0.5 dB [56]. Second, while the nominal frequencies of transmitter and receiver clocks are identical, they practically differ by a small Δf ; this problem is known as *carrier frequency offset*. Carrier frequency offset produces a continuous rotation of the received symbols. Practically, Δf is sufficiently small (e.g., $\sim 1\text{-}4$ KHz [60]), so that its effect is negligible over the CSS lengths/durations considered in this paper.

Hardware Implementation. The hardware implementation of CSS transmission and detection only requires the replication of components which are already present in off-the-shelf 802.11 chipsets, and specifically of filters and correlators. The basic implementation of 11ec needs m correlators (additional correlators may help increase channel reservations robustness as per Section 3.1.2). Because the correlated BPSK sequences at most require a sign flip on the received I/Q samples and several summations (with respect to expensive multiplications required to implement 802.11

floating-point correlators), CSS correlators occupy a very limited amount of resources.

3.3 Experimental Evaluation of 11ec

In this section, I present an experimental validation of 11ec using a measurement-based emulator that I design and implement. I perform the following experiments.

(i) I investigate the performance of 11ec in basic topologies that typically incur loss or imbalance for 802.11. **Our finding is that 11ec increases the throughput of under-served flows compared to 802.11 with or without RTS/CTS up to 9-fold (resp. 500-fold), with a benefit of almost 65% in fairness according to the Jain index.**

(ii) I investigate the performance of 11ec in a larger 5-flow network topology. **Our finding is that compared to 802.11 with or without RTS/CTS, 11ec achieves a gain of 46% (resp. 25%) in airtime utilization, and improves a severely underserved flow's throughput from 67 Kbps (resp. 0.4 Kbps) to 1.536 Mbps, for a gain of 2292%.**

(iii) I complete the evaluation of 11ec simulating its performance in a 20-node network topology. **Our finding is that 11ec considerably improves the performance of the under-served flows; for instance, the average throughput of the 70% lowest-throughput flows is 192% (resp. 632%) larger with 11ec than with 802.11 with (resp. without) RTS/CTS.**

The results in this section show that 11ec dramatically *improves network fairness*; furthermore, while such improvement can often decrease total utilization, 11ec remarkably *increases channel utilization*. Unlike 802.11, 11ec gives equal opportunities to weak links characterized by low data rates and strong high data rate links; for this reason, 11ec may sometimes achieve lower cumulative network throughput.

3.3.1 Measurement-driven Network Emulator

Our measurement-driven emulator is based on the GloMoSim simulator [72].* For the sake of realism, I modify GloMoSim in three ways: *(i)* I implement the support for multiple 802.11a/g modulations, i.e., BPSK 1/2, QPSK 1/2, 16QAM 1/2 and 64QAM 3/4 (corresponding to 6, 12, 24, 54 Mbps respectively); *(ii)* I implement a new propagation model that calculates links attenuation using my channel measurements; *(iii)* I implement a new traffic generator that draws the packet lengths from a distribution obtained experimentally. Specifically, with regard to the former, I perform a set of measurements at the channel emulator using the same transmitter/receiver/interferer setup described in the previous section, and I measure the BER as a function of the SINR. With regard to the second issue, I deploy up to 8 WARP nodes simultaneously in an office building (see Figure 3.7) and measure the signal strength between any pair, i.e., for any run of the experiment a single node transmits 400 μ s packets and all

*Despite that the last version of GloMoSim dates to late 2001, the basic operations of the 802.11 MAC layer are consistent with the latest standard. The physical layer includes features such as noise accumulation, which make it preferable to alternative simulators.

others record the received power. As a result of these two measurements, I manually select for each link in the network emulator the highest data rate that its channel SINR can support with negligible packet error probability. Finally, I integrate all results into my measurement-driven emulator. Next, I collect packet lengths of a user session running a mix of applications including web browsing, video downloading, audio streaming, and VoIP. The characteristics of the traffic are as follows: 773 different packet lengths, with a median value of 143 bytes, and average of 596 bytes. I implement a GloMoSim traffic generator that randomly determines the lengths of the transmitted packets following the distribution obtained experimentally.

CSS Implementation. I implement CSS reception and detection as an autonomous physical layer component, independent of the packet detection architecture of GloMoSim, i.e., CSS's are not simulated via small packets. Specifically, nodes store incoming CSS's, and schedule their evaluation after a delay corresponding to CSS's length, i.e., $6.35 \mu\text{s}$ for a 127-symbol CSS. For any stored CSS, the emulator keeps track of the variation of the background interference. At the moment of the evaluation the average SINR of the CSS is computed, and CSS detection is triggered if the SINR exceeds a threshold tuned to -6 dB for 127-symbol CSS (see Section 3.2.3). Our implementation permits each node to simultaneously store, evaluate and potentially detect multiple CSS's overlapping with other CSS's or incoming packets. Note that the original GloMoSim implementation of packet decoding does not support any of

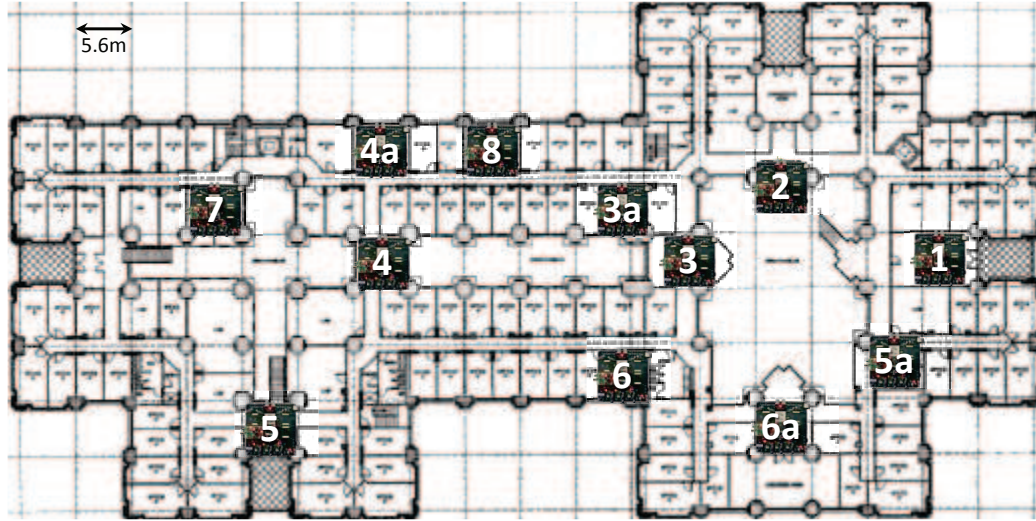


Figure 3.7 Layout of my office building deployment.

the features above, i.e., delayed evaluation and simultaneous multi-signal reception.

Finally, I implemented 11ec's MAC layer state machine by building on GloMoSim's 802.11. In particular, the design integrates the novel procedures corresponding, e.g., to deferral and timeout management.

3.3.2 Basic Topologies

In this set of experiments, I evaluate the performance of 11ec in a few basic topologies (mostly including two flows) that are characterized by symmetries or asymmetries in link signal strength differences and carrier sensing relationships. This study is important because these topologies are at the origin of throughput losses and/or imbalances in 802.11-based networks [10]. I show that 11ec largely overcomes the problems of 802.11 with and without RTS/CTS.

In my study, I classify the basic topologies into 4 main groups according to the prevalence of one of the two links with respect to the other (e.g., due to higher SINR), and to the carrier sensing relationships between the transmitters. Specifically, the 4 topologies are the following: (i) *Symmetric Hidden Terminals* include the case where two links are formed by transmitters that do not carrier sense each other, and share a common receiver; moreover, the links have similar reception power at the receiver. Specifically, these topologies include links whose signal strengths at the receiver are not different by more than 4 dB; I choose this threshold to exclude capture at any 802.11 modulation. (ii) *Asymmetric Hidden Terminals* include topologies where two non carrier sensing transmitters share a common receiver, but one of the formed links has a significant power advantage. Specifically, these topologies includes links whose signal strengths at the receiver differ by more than 5 dB; the choice of the threshold is to permit to one of the two links to capture over the other at BPSK modulation. (iii) *Information Asymmetries* include link pairs $a-b$ and $c-d$ whose transmitters a and c do not carrier sense each other, and whose receivers differ; moreover, one of the two links $c-d$ interferes with the other link $a-b$, but not viceversa. (iv) *Fully Connected WLANs* include topologies where all nodes carrier sense each other and transmit to a common receiver.

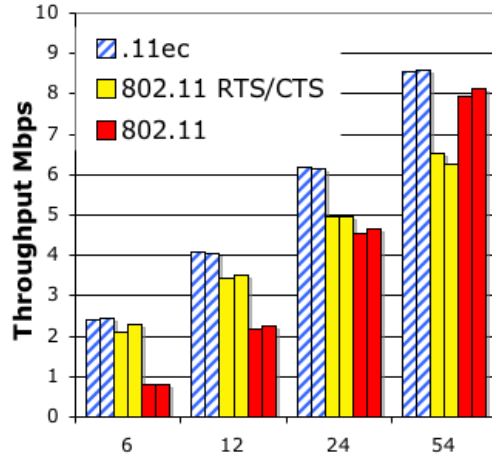
Most of the experiments in this section are performed by reproducing the topologies in my measurement-driven network emulator, with fully backlogged traffic and

packet lengths determined according to the experimental distribution described above. Each figure includes the bar graph of the throughputs of the flows for the three protocols I compare, namely 11ec, 802.11 with RTS/CTS, and 802.11 without RTS/CTS. Where utilized, RTS/CTS are transmitted at the OFDM 6 Mbps base rate. 11ec implementation includes CSS acknowledgements, but does not support RRTS mechanisms where not differently specified. The experiments in Figures 3.8(a) to 3.8(c) involve flow pairs; accordingly, the figures contain groups of six bars that correspond to the throughput of each flow, as achieved by the three protocols. The x-axes of the graphs indicate the data rates of the flows involved in the denoted experiment (when two different values are used, they orderly match the bar pairs), while the y-axes are in Mbps. In Figure 3.9 I introduce a metric termed *total airtime utilization*, which denotes the time share during which successful data packets are transmitted. Finally, I evaluate fairness according to two well known indicators, namely the Jain index [24], and proportional fairness [35]. The Jain index assumes values in the interval $[0, 1]$; for both indicators, higher values correspond to higher fairness.

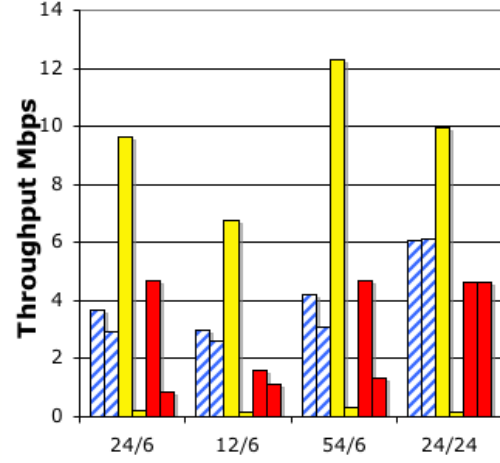
Symmetric Hidden Terminals. I consider fmy instances of symmetric hidden terminals: three of them are selected from my deployed network, and use modulation rates corresponding to 6/12/24 Mbps, respectively. The last case is artificially generated in order to explore the effect of using higher modulation schemes. Figure 3.8(a) shows that all solutions assign similar throughputs to both flows. However,

11ec and 802.11 with RTS/CTS achieve considerably higher total throughput than 802.11 for most rates. Furthermore, 11ec outperforms 802.11 with RTS/CTS due to the smaller duration of CSS's with respect to control messages (11ec control CSS exchange lasts $19.7 \mu s$, with respect to $128 \mu s$ of 802.11 control messages). Beside reducing the control overhead, this entails the decrease of the collision probability (see Section 3.1.4) from 13% for 802.11 with RTS/CTS to 6.5% for 11ec irrespective of the rate used to transmit the actual data. The effect of the overhead reduction becomes more and more evident as the packet data-rate increases; at 54 Mbps the total throughput gain of 11ec over 802.11 with RTS/CTS is about 34%. Finally, at 54 Mbps the data packets are sufficiently short to permit low collision probability to 802.11 without RTS/CTS (19% as opposed to 66% obtained at 6 Mbps); nonetheless, 11ec still shows 5% throughput gain.

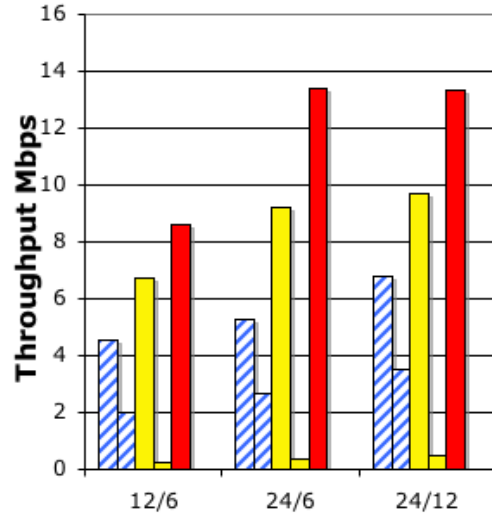
Asymmetric Hidden Terminals. I consider fmy instances of asymmetric hidden terminals, all based on actual link power measurements. In these topologies, packets sent at base rate (e.g., control packets) by the sender with high SNR capture over packets sent by the sender with low SNR. However, because of my choice of the data modulation rate as the highest that can be supported by the link in the absence of interference, data packets always collide for both senders. Figure 3.8(b) shows that the capture effect has disastrous consequences for the flow with low SNR in 802.11 with and without RTS/CTS, while it has no effect on 11ec. For example, in



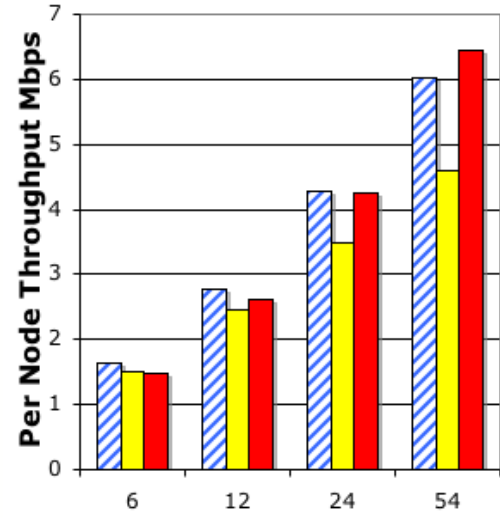
(a) Sym Hidden Terminals



(b) Asym Hidden Terminals



(c) Information Asymmetry



(d) Fully Connected WLANs

Figure 3.8 Throughput of 11ec, 802.11 with/without RTS/CTS in basic topologies

the first instance represented (i.e., the left-most six bars in the figure) 11ec improves the throughput of the under-served flow by about 13-fold (resp. by about 2.5-fold) with respect to 802.11 with (resp. without) RTS/CTS. In 802.11 with RTS/CTS the imbalance is due to the fact that in case of overlapping RTS, the RTS of the stronger link is correctly decoded, while the other is ignored; in all instances the RTS collision probability of the weaker link is over 50%. A similar consideration is in order for 802.11ec; in fact, link SNR's can be sufficiently different to permit the CSS's from one of the links to capture over the CSS's from the other in case of overlapping. However, the probability of CSS's overlapping is so low for 802.11ec, that even in such cases (not shown in the figure) the resulting throughput imbalance remains confined within about 15%. Since 802.11 without RTS/CTS does not use the base rate, but transmits all packets at data rate, the throughput imbalance is originated only by the shorter duration of the data packet of the dominant link, which permits it to enjoy higher success probability. The packet collision probability of the weaker link is again over 49% for all cases of heterogeneous rates. The result for the last instance (i.e., last two bars in the graph) supports this claim: when both links operate at 24 Mbps, their throughputs do not depend on any SNR imbalance, and the collision probability is reduced to 30%. Finally, because of the larger number of data packet collisions (that have a longer duration than RTS or CSS collisions), 802.11 without RTS/CTS has a lower total throughput. In contrast to 802.11, 11ec reduces the imbalance by reducing

the packet loss to about 11%.

In terms of fairness, considering for example the first instance, 11ec improves the Jain index from approximately 0.52 and 0.68 in 802.11 with and without RTS/CTS to 0.99; similarly, in terms of proportional fairness, 11ec improves the sum of the logs of the share of the rates from -1.67 and -0.87 of 802.11 with and without RTS/CTS to -0.6. The total throughput of 11ec is lower than the competitors; however, this is misleading, and is due to the fact that 11ec improves the throughput of flows with lower data rate. Figure 3.9 clarifies this aspect, by showing the total airtime utilization for all instances represented in Figure 3.8(b). 11ec obtains up to 30% higher airtime utilization than the other 802.11 versions.

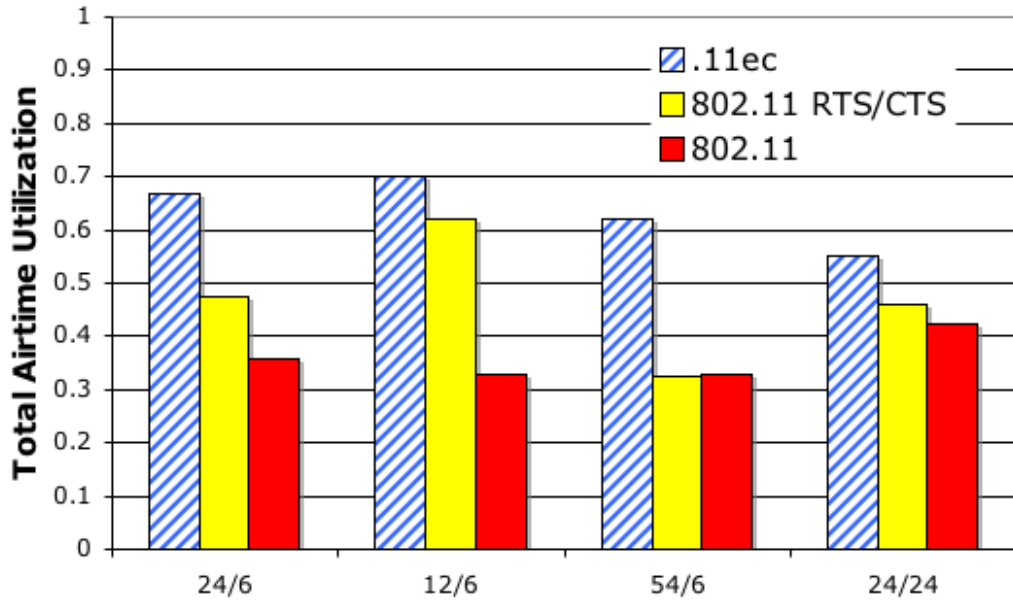


Figure 3.9 Total airtime utilization in the case of Asymmetric Hidden Terminals.

Information Asymmetry. The three instances I consider are again based on my channel measurements. In these topologies, the link interfering always succeeds, while the interfered may become severely under-served due to high number of collisions. In fact, the sender of the under-served link cannot perceive when the channel is free at its receiver, and randomly selects transmission instants; in the likely case of collision, the sender of the under-served link decreases its sending rate due to backoff. Figure 3.8(c) shows that the information asymmetry completely starves the under-served link in 802.11. 802.11 with RTS/CTS slightly outperforms the version without, due to the larger probability of the under-served flow to correctly transmit an entire RTS without being interrupted by the interferer. In fact, the collision probability decreases from almost 100% for packets transmitted by 802.11 without RTS/CTS to 80-90% for RTS transmitted by 802.11 with RTS/CTS. It is important to notice that several of the bars corresponding to 802.11 flows in this figure are almost or completely invisible. In contrast, even without the feature of control during data, 11ec manages to assign a significant throughput (about 45% of the interfering link) to the under-served link. For example, in the last instance represented (i.e., the left-most six bars in the figure) 11ec improves the throughput of the under-served link by 9-fold (resp. by 500-fold) with respect to 802.11 with (resp. without) RTS/CTS. In terms of fairness, 11ec improves the Jain index from approximately 0.5 in 802.11 with and without RTS/CTS to 0.87, with a 65% gain, and proportional fairness from -1.47

and -3.34 of 802.11 with and without RTS/CTS resp. to -0.67. Similarly to the case of asymmetric hidden terminals, 11ec achieves a cumulative throughput lower than 802.11 but gains up to 21% and 8.5% in airtime utilization with respect to 802.11 with and without RTS/CTS respectively. These results are due to the small size of control CSS's that have a high probability of being received during free channel intervals.

Fully Connected WLANs. Figure 3.8(d) shows the average throughput obtained for three carrier-sensing links, and groups of three bars correspond to the three protocols compared. In this scenario, 802.11 without RTS/CTS generally performs the best due to low overhead and rare collisions. In the worst case of 54 Mbps, 11ec obtains 6.9% less throughput than 802.11 without RTS/CTS, while 802.11 with RTS/CTS achieves 23% less throughput than 11ec due the control message overhead. At low data rates all protocols perform similarly due to the long packet durations. This result clearly shows that even in the absence of hidden terminals, control CSS's and larger slot size of 11ec do not involve significant throughput penalties.

Final Remarks. First, in the experiments above, I consider UDP traffic. I notice that the throughput imbalances I observed for 802.11 would negatively affect TCP dynamics. By inspecting the traces, I also notice that even in cases of balanced throughput (such as *symmetric hidden terminals*), 802.11 (with and without RTS/CTS) alternately serves for long periods of time one of the two links, by almost starving the other [5]; this is extremely detrimental for TCP performance. Second,

my experiments do not implement rate-adaptation, but manually select the best rate achievable based on the links SNR. I observe that rate adaptation does not produce any benefit to 802.11 with RTS/CTS, since control messages need still be transmitted at base rate, and data packets rarely collide. On the other hand, 802.11 without RTS/CTS may benefit in case of hidden terminals, but typically at the price of higher unfairness even in fully connected WLANs due to capture effect [46]. Finally, it is remarkable to notice that in contrast to common tenets of related literature, RTS/CTS at 6 Mbps does produce a large performance improvement vs. without RTS/CTS, and only slightly penalizes the throughput in the absence of hidden terminals.

3.3.3 Network Wide Emulation

Here I investigate larger topologies in order to demonstrate the fairness gains of 11ec in case of multiple flow interactions. I consider a 5-flow topology based on the channel measurements I performed; the flows operate at 24, 24, 24, 54, and 6 Mbps, respectively, and each one conveys fully backlogged traffic, with packet length distribution as discussed above. Figure 3.10 shows the detailed bar graph of the throughput of all flows for 11ec and 802.11 with/without RTS/CTS; the flows referred on the x-axis match the node locations in Figure 3.7. The figure shows that 802.11 with RTS/CTS and 11ec achieve higher fairness than 802.11 without RTS/CTS. In addition, while 802.11 with RTS/CTS almost starves flow $4a \rightarrow 3a$ by assigning 67

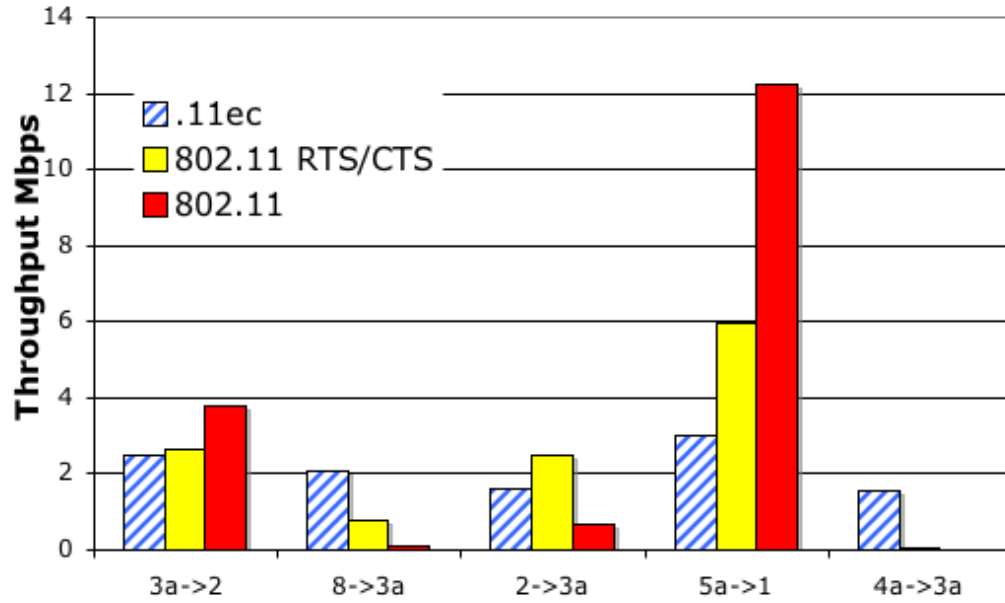


Figure 3.10 Throughput distribution for a 5-flow topology.

Kbps, 11ec manages to assign it 1.536 Mbps for a gain of 2292%; 802.11 without RTS/CTS only assigns 0.4 Kbps to that flow. Because the flow operates at 6 Mbps, this has a large effect on the airtime utilization, which increases from 0.41 and 0.48 of 802.11 with and without RTS/CTS to 0.60 of 11ec, for a gain of 46% and 25% respectively. Finally, 11ec significantly increases throughput fairness; specifically, the Jain index is 0.57, 0.34, 0.93 for 802.11 with and without RTS/CTS, and 11ec respectively. 11ec shows about 30% higher proportional fairness with respect to 802.11 with RTS/CTS.

3.3.4 Large Network Simulation

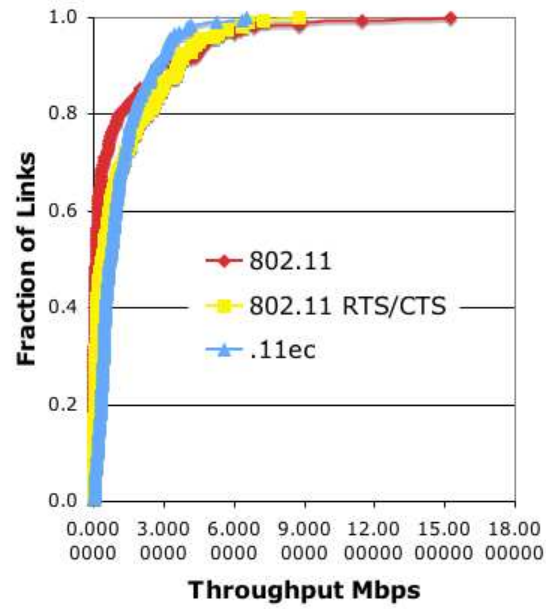
This experiment aims to evaluate the performance gain of 11ec in larger network topologies including 20 nodes. The scenario is obtained by randomly deploying 20 nodes within an area of $250m \times 250m$, and modeling the channel according to a path-loss with attenuation exponent of 3.5; the nodes use a transmission power of 100 mW. The resulting link SNRs dictate the data-rate that can be utilized on the links, and their carrier sensing relationships; according to the latter, the network spans at most 5 hops. In this experiment, the link data-rates are manually selected in order to achieve reliable decoding for an interference 3 dB stronger than noise, i.e., in order to maintain an SNR margin of 3 dB. For each node, the data flow is determined by randomly choosing one of its neighbors as the destination; the traffic is fully backlogged, and the packet lengths are chosen according to the experimental distribution above. Figure 3.11(a) shows the cumulative distribution function (CDF) of the link throughputs achieved by the three protocols; Figure 3.11(b) is a zoomed version as described below. In the figures, the y-axes represent the fraction of flows, while the x-axes throughput values (in Mbps). Accordingly, the plotted values indicate the fraction of flows with a throughput smaller or equal to the corresponding abscissa; e.g., a point in (0.4,0.444) means that 40% of the flows have a throughput lower or equal than 444 Kbps. Notice that the flows are represented in the graphs according to the sorted values of their throughputs, i.e., neither the x-axis nor the y-axis are related to the

specific flow identities.

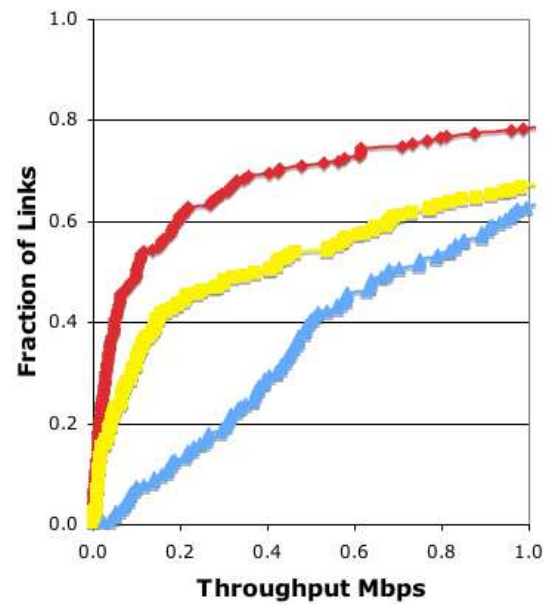
The left plot represents the entire CDF, while the right plot magnifies the results for the flows that achieve less than 1 Mbps. Clearly, the right plot shows that 11ec highly benefits the under-served links, i.e., the bottom 70.5% (resp. 85.5%) with respect to 802.11 with (resp. without) RTS/CTS. Specifically, the average throughput of the 70% lowest-throughput flows is 192% (resp. 632%) larger for 11ec than for 802.11 with (resp. without) RTS/CTS. Moreover, at the 20/40/60 percentiles 11ec achieves a throughput value 7.7x/3.5x/1.4x (resp. 25x/10x/4.8x) larger than 802.11 with (resp. without) RTS/CTS. As discussed above, the improvement in 11ec comes at the price of reduction of the high-throughput flows; in fact, the left plot shows that the top 9%-29% flows achieve best performance with 802.11 with RTS/CTS, while top 9% obtain highest throughput with 802.11 without RTS/CTS. In terms of airtime, 802.11 with (resp. without) RTS/CTS achieve 0.61 (resp. 0.39), while 11ec achieves 0.76. The Jain index improves from 0.34 (resp. 0.27) for 802.11 with (resp. without) RTS/CTS to 0.53.

3.3.5 Extensions: Control During Data Simulation

In Section 3.1.1 I introduce the possibility of conveying control information during data reception or overhearing, by leveraging the robustness of CSS's. This technique is particularly useful when the SINR on a weak link is -6 dB or more with respect



(a) Entire CDF



(b) Zoom in

Figure 3.11 Throughput of 11ec, 802.11 with/without RTS/CTS in 20-node simulated topologies

to the hidden terminal interferers. Consider for instance the *information asymmetry* topology; the major issue is that the sender of the weak link cannot determine the state of the medium at its receiver. Consequently, the transmissions of the sender likely overlap with the interferer transmissions and cannot be decoded. Instead, in 11ec the receiver can detect the initiation CSS's and be notified of the sender's intention to communicate even while the interferer is simultaneously active, as long as the SINR conditions allow it. When the transmission of the interferer is over, the receiver may contend for the medium in spite of the sender, and transmit an RRTS that prompts the sender to perform the data exchange, while preventing the interferer from accessing the medium.

I implemented this mechanism as a variation of 11ec and verified its performance in information asymmetry topologies, designed to satisfy the SINR condition above. In the experiment, the weak link operates at 6 Mbps, while the data rate of the interfering link varies in the range of 6, 12, 24, 54 Mbps in the different experiments. Figure 3.12 contrasts the throughput on the two links for 11ec with and without the RRTS enhancement for the different data-rate configurations. The figure shows that in this case RRTS highly improves the performance of the weak link with respect to the interfering, and even leads the weak link to obtain the higher throughput. This is due to the fact that as soon as the medium becomes interference free, the receiver reserves it for the sender. In general, the results show a fairer throughput

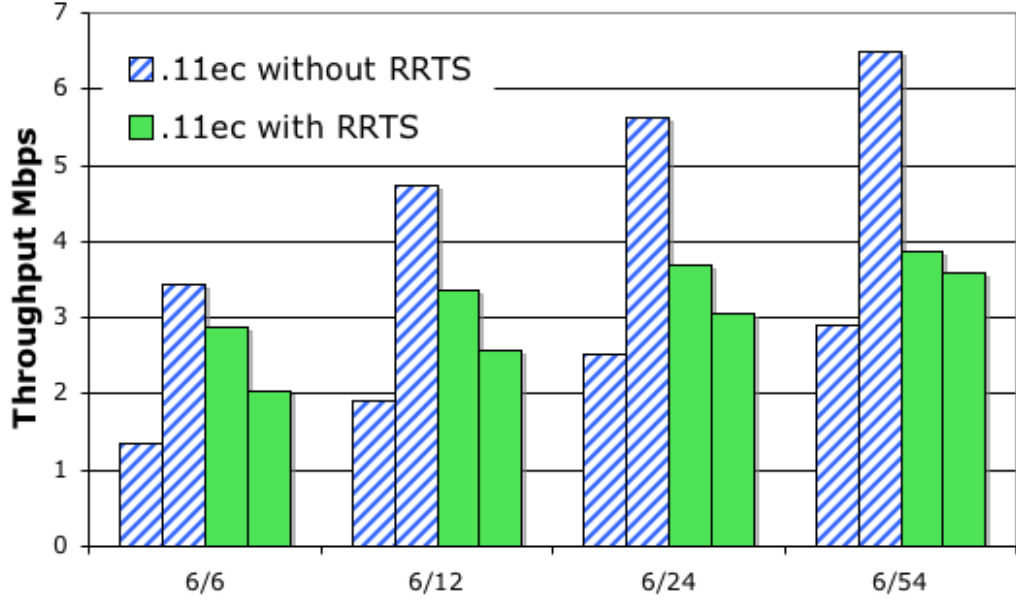


Figure 3.12 Effects of the use of the feature of Control during Data in 11ec on Information Asymmetry topologies.

distribution between the links. Specifically, the Jain index improves from 0.84-0.87 to 0.97-0.998, while the proportional fairness is increased of 14%. The increase in fairness is accompanied by an increase in airtime utilization ranging from 2% to 10%.

3.A Signal Correlation

A CSS s is detected via cross-correlation with a local copy, i.e., at the reception of a complex signal y that may contain s , y is cross-correlated with the complex conjugate of the target CSS s^* . Formally, for a CSS of length L ,

$$C(\Delta) = \sum_{k=0}^{L-1} s^*(k)y(k + \Delta) \quad (3.2)$$

where Δ represents the position of the correlation with respect to the input signal, i.e., the sample for which we perform the correlation. Note that: (i) if $[y(\Delta) \dots y(\Delta + L - 1)]$ does not contain exactly s , the value of $C(\Delta)$ is nearly 0; (ii) if $[y(\Delta) \dots y(\Delta + L - 1)]$ contains a copy of s with sufficient SINR, the $C(\Delta)$ obtains a large value proportional to the energy of the signal.

We use cross-correlation as a test statistic for the detection of a target CSS, and repeat its computation at each new sample of the incoming signal. Specifically, detection is performed by setting a threshold T (see Section 3.2.6); if $C(\Delta) \geq T$ (resp. $C(\Delta) < T$), the presence (resp. absence) of the CSS in $y(k + \Delta)$ is declared. The performance of cross-correlation can be quantified in terms of false positives and false negatives. Specifically, after deciding on a threshold T , a false positive is declared when $C(\Delta) < T$ even though the CSS is present within $[y(\Delta) \dots y(\Delta + L - 1)]$, and a false negative is declared when $C(\Delta) \geq T$ even though the CSS is absent. Cross-correlation detection provides a processing gain, which is linear in the length of the correlated sequence [34]. This means that a sequence of length $2L$ obtains a value of $C(\Delta)$ twice as large a sequence of length L , i.e., it is considerably more likely to exceed T .

Chapter 4

MIDAS

4.1 Introduction

Managed enterprise WLANs and wireless mesh networks regularly encounter underperforming links, i.e., links with throughput below an acceptable value determined by the operator. A key corrective action available to the network manager is to throttle other nodes that may be hindering the underperforming link. However, to do so first requires identifying which node to throttle. While it is clear it should be a “neighbor,” there may be a large set of candidate nodes for which throttling can have vastly different effects, including no effect on the under-served link. Moreover, it is not immediately evident how much throttling any node will increase the throughput of the targeted under-served link due to complex node interactions and coordination.*

In this paper, I design MIDAS, a framework that uses online measurements of network performance to infer the most hindering nodes which cause a target link to be under-served or to obtain poor performance. Moreover, MIDAS identifies effective management actions to increase the performance of the under-served link by appropriately limiting the transmission rates of the hindering nodes. Finally, I implement MIDAS on real hardware and investigate its performance in an indoor testbed and

*Network managers have a number of options for mitigation, including moving sets of APs or clients to alternate frequencies. MIDAS could equally be applied to such strategies (it would identify the best ones to move and could recompute the new throughputs). However, evaluation of such alternate mitigation schemes is beyond the scope of this paper.

simulation.

MIDAS employs a methodology comprising three procedures: *i) measurement collection*, which gathers reports from each node consisting of a small set of passive time-aggregate measurements; *ii) inference*, which infers the coordination among the transmissions of different sets of nodes using the reported measurements; *iii) prediction*, which utilizes the inferred information to predict the throughput gain of any target link, corresponding to rate-limiting different conflicting nodes. In particular, my contributions are as follows.

First, I introduce the concept of *Activity Share* which characterizes the coordination and interference among any set of conflicting nodes. The throughput of a link is influenced by the sender busy time (i.e., the more the sender senses the medium busy the less it can transmit), and the collision probability (i.e., even if it can transmit, its transmissions are corrupted). Coordination is critical to understand how different nodes contribute to busy time and collision probability of each other. In fact, a sender's busy time is not simply the sum of the transmission times of its neighbors, as neighbors which are hidden from one another may transmit simultaneously. Analogously, link collisions are not the sum of the collisions with *each* hidden terminal, because multiple hidden terminals may collide with the same packet. Therefore, knowing how much conflicting nodes (i.e., neighbors of sender or receiver of a link) are destructive to the link requires understanding their coordination. In order to capture

node coordination, I define network state as a set of transmitting nodes; accordingly, in each time instant the network is in a unique state. I define *Activity Share* as the time share the network spends in each possible state in a given interval. That is, the Activity Share is a vector including, for each possible set of nodes, the fraction of time they spend transmitting simultaneously. Note that the Activity Share depends not only on the topological relationships between the nodes as determined by carrier sensing and link interference, but also on the transmission rate and pattern of each node under the current traffic conditions. Furthermore, since the transmission pattern of any node depends on the transmission pattern of its neighbors, and the transmission pattern of its neighbors depends on the transmission pattern of their neighbors (and thus recursively of all nodes in the network), the Activity Share captures the effects of global network interactions that extend beyond node locality. In particular, the Activity Share captures the coordination among the transmissions of any set of conflicting nodes as determined by the current global network conditions. In contrast, alternative indicators, such as the individual node transmission rates, are insufficient to determine how conflicting nodes influence the target link, since they do not capture the coordination. For example, the conflicting node with the highest transmission rate might mostly transmit simultaneously with other conflicting nodes, such that limiting its rate may scarcely benefit the target link. I will show how the manager can utilize the Activity Share as a tool to understand the network behavior

and to determine a strategy to change it, e.g., to increase the throughput of a congested or underperforming link. Unfortunately, the estimation of the Activity Share is challenging, because it cannot be locally measured by the nodes. In fact, during the reception of multiple overlapping packets, nodes cannot identify all senders, and thus recognize the network state.

Second, I design a tool to infer the Activity Share using a small set of *passively collected, time-aggregate* local channel measurements, reported by every node. Inferring the Activity Share requires computing the temporal distribution of the different network states, i.e., how long the network spent in each of them. I develop a technique to eliminate infeasible distributions by incorporating physical rules (e.g., the busy time of a node should coincide with the sum of the durations of the states in which its neighbors transmit and that node does not). Unfortunately, there can be an infinite number of temporal distributions that yield identical measurements. Consequently, I penalize unlikely distributions by incorporating protocol rules (e.g., the occurrence of states in which adjacent nodes simultaneously transmit is unlikely), and select a representative by using a statistical approach based on entropy considerations. To further limit the complexity of my problem, I propose a technique to reduce its dimensions, by actually *eliminating* the unlikely distributions.

Third, I develop a tool to predict the throughput increase achievable on the target link by rate-limiting the links formed by the target link's conflicting nodes. The

Activity Share permits assessment of the current network conditions; however, it lacks predictive power to identify effective rate-limiting actions and to anticipate their outcomes. The challenge is to understand how changing the transmission time of a conflicting node affects the Activity Share, and subsequently how the new Activity Share affects the target link's throughput. I design a simple throughput prediction model that derives its inputs from the current network conditions, i.e., from the inferred Activity Share, thus representing the first throughput model based on online measurements.

Fourth, I extensively evaluate the accuracy of MIDAS by combining testbed experiments and simulations. I implemented MIDAS on real hardware and deployed it on an indoor testbed, where I investigated its sensitivity to different network settings under real channel conditions. The results show that MIDAS infers the Activity Share with high accuracy, i.e., with a mean relative error as low as 4%. In order to extend my validation to a broader set of scenarios, we performed numerous simulations. A key finding is that, by rate-limiting different conflicting nodes for the same fixed amount, the throughput of the target link can increase from 7% to 172% of the rate-limited quantity. I also validate the effectiveness of the Activity Share in supporting throughput prediction, and show that MIDAS anticipates the benefits of alternative rate-limiting actions with an error lower than 20% of the rate-limited quantity.

The remainder of the paper is organized as follows. In Section 4.2 I present MIDAS, and define the Activity Share. I develop a technique to infer the Activity Share in Section 4.3. A throughput prediction tool using the Activity Share is described in Section 4.4. Section 4.5 presents testbed and simulation results.

4.2 The MIDAS Framework

A link can be considered under-served due to a discrepancy between the network manager’s targeted link throughput and the actual throughput. The network manager’s policy for setting target throughputs (incorporating factors such as fairness, QoS, pricing, offered load, etc.) is beyond the scope of this paper. The objective of MIDAS is to determine the causes of the poor performance and design corrective actions.* While local node observations can point out problematic links, in general the causes of the low throughput cannot be locally inferred. For instance, in the case of high packet drop rate, the local measurements can seldom determine the hindering nodes. MIDAS helps improving the problematic link by inferring the impact of hindering transmitters and by rate-limiting the most destructive flows.

The severity of link hindering interactions mainly depends on three factors: *i)* network topology: nodes’ pairwise relations, as determined by carrier sensing and interference, determine the form of interaction, e.g., hidden terminals are responsi-

*In this paper, I only consider 802.11 MAC issues, e.g., I do not address throughput losses due to TCP dynamics, or low received signal strength.

ble for transmission corruptions, while carrier sensed nodes affect deferral; *ii*) link transmission rates: nodes that transmit few packets are less likely to interfere with a target link; and *iii*) link transmission coordination: the number of packets transmitted on a link and corrupted by a hidden terminal depends on how frequently the link sender and hidden terminal transmit simultaneously. Note that transmission rates and coordination strongly depend on the traffic load of each node.

In this section, I introduce a novel metric termed *Activity Share* which captures the coordination between any possible set of nodes, by measuring the fraction of time they transmit simultaneously. Even though the Activity Share does not directly measure the interference between nodes, it reflects node interactions. Thus, the Activity Share is affected by node topological relationships, traffic load, MAC protocol, etc. I will show how MIDAS can utilize the Activity Share to evaluate the potential effects of alternative corrective actions (see Section 4.4). I will also show that the Activity Share cannot be locally observed by the network nodes and describe how it can be inferred from measurements collected by the nodes. Note that, in contrast to the Activity Share, alternative indicators that evaluate *pairwise* conflicts between interfering links taking into consideration *only topological* information (e.g., the conflict graph [25]) miss the important dynamic information about the coordination of the transmissions of multiple nodes.

4.2.1 The Activity Share: Fundamental Element of Network Observation

As previously explained, my management framework aims to identify the originating causes of under-served links and to increase their throughput by rate-limiting conflicting nodes. In this study, I consider 802.11 stationary multi-hop wireless networks, including enterprise WLANs and mesh networks. In such networks, nodes can affect the throughput attained on a link (sender-receiver pair) by two key means: *i*) reducing the time the medium is perceived as free by the sender, thereby forcing the sender to defer; *ii*) corrupting the packet reception at the receiver end, i.e., colliding. In multi-hop topologies, despite the use of the carrier sensing mechanism, several nodes that are in conflict with a specific transmitter can potentially transmit simultaneously. Hence, it is challenging to anticipate the benefits of rate-limiting conflicting links on the sender busy time or collisions of the target link, and thus on its throughput. Even knowing the exact packet transmission rate of each node in conflict with the link of interest is not sufficient, because the throughput gain mainly depends on the coordination among the conflicting nodes as illustrated in the example below.

Example. The following example shows that node coordination is the key to understand the effectiveness of rate-limiting conflicting nodes to improve the throughput of an under-served link. Let us consider the simple wireless network depicted in Figure 4.1(a), where a dotted line connecting two nodes indicates that the two nodes are within carrier sensing range. The link (a, b) is identified as under-served; the goal

of the network manager is to assess how decrementing the transmission rates of the conflicting links formed by nodes 1 and 2 can benefit the throughput of link (a, b) . Since nodes 1 and 2 are not coordinated by carrier sensing, they can transmit simultaneously. Figure 4.1(b) depicts a typical timeline of the transmissions of the three nodes. The continuous deferral is the cause of the performance issue of link (a, b) ; in fact, (a, b) can transmit only when both nodes 1 *and* 2 are silent. Thus, decreasing the transmission rate of only one of them will produce a minimal benefit to (a, b) ; this is because only a small portion of the released airtime will result in free airtime for (a, b) . The analysis of the coordination between the conflicting nodes 1 and 2, and in particular of the large overlap between their transmissions, can promptly lead to this conclusion. Obviously, this is only a simple case, where the large overlap between the transmissions of 1 and 2 is not surprising; however, in more complex topologies with several conflicting nodes, it is not clear how to determine node coordination and its effect.

Network State and Activity Share. The key to understanding how conflicting nodes affect an underserved link is to determine the time they spend transmitting simultaneously. For instance, in the example in Figure 4.1, the transmissions of nodes 1 and 2 mostly overlap, anticipating a small gain in free airtime perceived by the link (a, b) , from the reduction of the transmission times of either one. Furthermore, the higher the number of nodes in conflict with a target link which can potentially

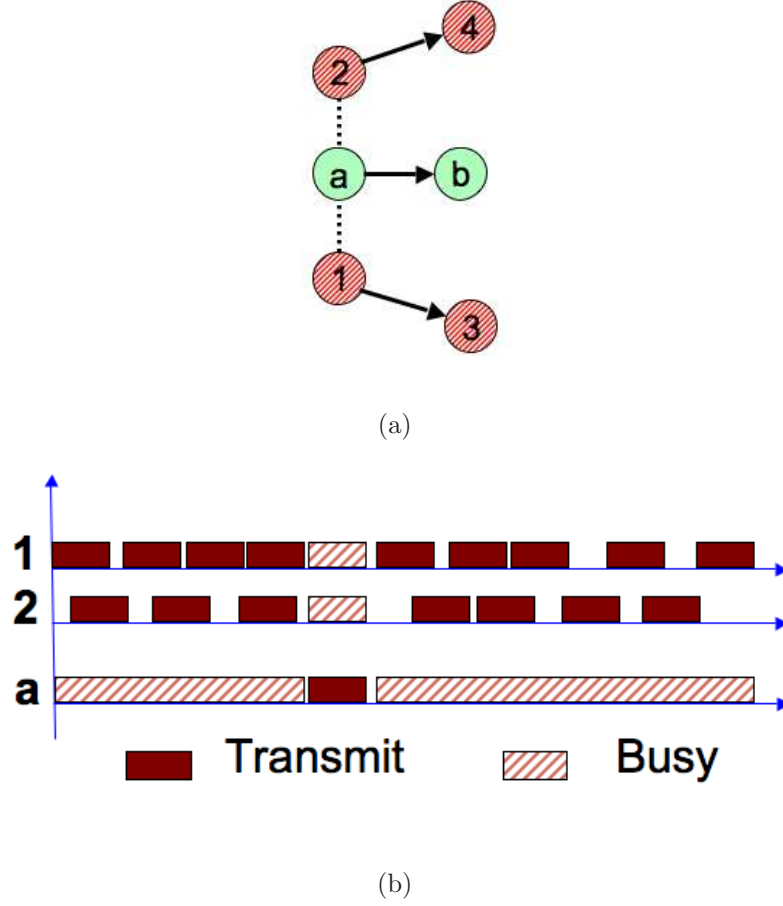


Figure 4.1 Example of transmission alignment due to (lack of) carrier sensing.

transmit simultaneously, the lower the gain from limiting the transmission time of a single node. For instance, if in the example instead of two uncoordinated nodes in conflict with link (a, b) , there were three or more such nodes, the free airtime gained by rate limiting a single node would be even lower.

Let us consider an N -node network. To formalize the concept of simultaneous transmission of a set of nodes, I define *network state* and *Activity Share* as follows.

Definition 1 The Network State \vec{D} denotes the transmission status of each node in

the network. \vec{D} is an N -dimensional vector comprising an entry for each node that indicates whether the node is transmitting or idle in the state. $\vec{D} = (d_1, d_2, \dots, d_N)$, $d_i \in \{0, 1\}$, where $d_i = 1, 0$ indicates that node i is transmitting or not, respectively. Note that each network state is univocally identified by the set of transmitting nodes.

Since there are N nodes in the network there are 2^N possible states denoted by $\vec{D}_1, \vec{D}_2, \dots, \vec{D}_{2^N}$. The network transitions in time through a succession of network states. The *Instantaneous Network State* at time t_0 , $\vec{D}(t_0)$, is the state of the network at time t_0 , i.e., $\vec{D}(t_0) = \vec{D}_j$ iff the network state at time t_0 is \vec{D}_j .

Next, I define the *Activity Share* which is the time share the network spends in each state per time unit.

Definition 2 The Activity Share of the network state \vec{D}_j , denoted by $AS(L, \vec{D}_j)$, is the fraction of time during the interval $[0, L]$ for which the network was in state \vec{D}_j , i.e., $AS(L, \vec{D}_j) = \frac{1}{L} \int_0^L \mathbf{1}_{[\vec{D}(t)=\vec{D}_j]}(t) dt$, where $\mathbf{1}_{[\vec{D}(t)=\vec{D}_j]}(t)$ denotes the indicator function such that $\mathbf{1}_{[\vec{D}(t)=\vec{D}_j]}(t) = 1$ if the network state at time t is \vec{D}_j , and 0 otherwise. The sum of $AS(L, \vec{D}_j)$ over all possible states adds to one:

$$\sum_{j=1}^{2^N} AS(L, \vec{D}_j) = 1 \quad \forall L \quad (4.1)$$

I separately denote as Activity Share, \vec{AS} , the distribution of time the network spent in each state during the time interval $[0, L]$, i.e., $\vec{AS} = \{AS(L, \vec{D}_j), \forall \vec{D}_j\}$. Note that if the network is stationary $\lim_{L \rightarrow \infty} AS(L, \vec{D}_j)$ is the probability that the network at

any time instant is in state \vec{D}_j . In the following, I consider L large enough to satisfy stationarity, and I drop L from my notation.

The estimation of the Activity Share is challenging because it cannot be locally measured by the nodes. Specifically, the nodes cannot identify the transmitters of all the packets they carrier-sense. In fact, some of the overlapping packets (e.g., sent by 1 and 2 in Figure 4.1) may collide at the intermediate nodes (e.g., node a), preventing the decoding of at least one of them. Another obstacle is the strength of the received signal, which may exceed the carrier sense threshold, but not be sufficiently greater than the background noise (plus interference) to permit the decoding of the packet. In order to overcome these challenges, it is necessary to analyze the combined measurements of different nodes.

4.2.2 The Measurements

In MIDAS, each network node k continuously collects information, and delivers a report R_k to the manager at every report interval.* In this paper, I suggest a new scheme which I will use to infer the Activity Share, given a set of measurements reported by the nodes $\vec{R} = \{R_1, R_2, \dots, R_N\}$.

A tradeoff emerges between the amount of information contained in R_k and the

*The actual placement of the manager's node is an important design parameter; in fact, an optimal placement would permit to reduce MIDAS overhead by efficiently aggregating node reports, and to shorten the report delivery delays. However, in case the manager's node is not optimally placed, MIDAS reports can still be delivered leveraging regular wireless network routing, with minimal overhead penalty due to the reports' small size.

estimation accuracy of the Activity Share. If R_k contained complete traces of the exact times and durations of all transmissions of node k , the manager could use the reports to reconstruct a global trace of the transmissions in the network (such as in Figure 4.1(b)), and hence obtain the Activity Share by inspection. However, the amount of information that needs to be collected and the timely delivery of such traces would overwhelm the network resources. For example, a set of traces satisfying my requirements is collected in [12]; therein, the authors show that the overhead is between 100 kbps and 500 kbps per node, without even considering the multiplicative effect of multi-hopping [9].

I consider a highly simplified and easily measured set of inputs R_k consisting of information passively collected from the local network card and time averaged over the report interval. Each node observes the local channel in three states: T if the measuring node is transmitting; B if the node is not transmitting but the total received energy exceeds the carrier sensing threshold; I if neither the received energy exceeds the carrier sensing threshold, nor the node itself is transmitting. Notice that the state B reflects the activity of all carrier sensed nodes and does not distinguish between different transmitters. The report R_k includes the time shares T_k , B_k , I_k node k observed the channel in any of the three states during the report interval. Clearly, $T_k + B_k + I_k = 1, \forall k$; thus, R_k needs to include only two out of the three time shares. An implementation of the measurement collection tool is presented in Section

4.5.1. In contrast to trace-based solutions, my reports only include two numerical values.

4.3 The Inference Tool

The reconstruction of the Activity Share from the reports is challenging because the time-average measurements in R_k are the result of the transmissions either of the individual node k (i.e., T_k) or of *all* its neighbors (i.e., B_k). In both cases, it is not possible to locally determine the overlapping intervals of subsets of neighbors, and of sets of nodes that do not share neighbors. In this section, I will show how to overcome this issue; my solution consists of three elements. First, in order to obtain accurate estimations, I use the \vec{R} inputs to constrain the domain of the feasible \vec{AS} (Section 4.3.2). Since the constraints do not generally identify a unique solution, I propose an optimization problem to choose a single representative \vec{AS} (Section 4.3.3). The last element of the solution addresses the computational complexity of the proposed problem, and reduces the dimension of the \vec{AS} solution space using protocol rules of 802.11 (Section 4.3.4). In the experimental results in Section 4.5 I consider practical implementation issues, such as report losses and time-varying channel.

4.3.1 Network Model

I consider a single-radio, single-channel network, and I abstract it as a graph $G = \{V, E\}$, where the vertices V represent the N nodes, and the edges E represent the

carrier sensing relationships among the nodes. The existence of a sensing edge $(i, j) \in E$ means that node i carrier senses transmissions from node j and vice versa. I define the set of the nodes that node i carrier senses as $V_{cs}(i) = \{j | (i, j) \in E\}$. I assume that the topology of the graph with respect to E is fixed during any observation interval and known to my inference tool (e.g., via offline link profiling [54], or passive online estimations [32]).

4.3.2 Report-based Constraints

In order to obtain an accurate estimation of the Activity Share, I use the reported measurements \vec{R} to constrain the feasible domain. Since the local observations of the channel of any node provide information about the cumulative duration of sets of network states, the actual \vec{AS} must satisfy the constraints imposed by all local observations, and hence lies in the feasible region the observations define. Accordingly, I can derive the following constraints:

$$\sum_{j:(D_j^k=1)} AS(\vec{D}_j) = T_k \quad (4.2)$$

$$\sum_{j:(D_j^k=0) \wedge (\exists s \in V_{cs}(k): D_j^s=1)} AS(\vec{D}_j) = B_k \quad (4.3)$$

$$\sum_{j:(D_j^k=0) \wedge (D_j^s=0, \forall s \in V_{cs}(k))} AS(\vec{D}_j) = I_k \quad (4.4)$$

$$\forall k \in [0..N]$$

where D_j^n denotes the n -th component of the \vec{D}_j vector. Equation (4.2) constraints the time share each node is transmitting: the sum of the Activity Shares of states in which node k transmits should be equal to the fraction of time k transmitted. Equation (4.3) is related to the busy time of the nodes. In my network model, the state of a node k is busy if the node is not transmitting and any of the nodes in $V_{cs}(k)$ is transmitting. Hence, the Activity Shares of states, in which any of the nodes in $V_{cs}(k)$ is transmitting and node k is not, sum up to B_k . Notably, also the busy time of the nodes carries information about the Activity Share, by inducing constraints on the duration of the network states including transmissions from any neighboring node. Equation (4.4) relates to the idle time of the nodes, and can be obtained with considerations analogous to the previous two. Simple considerations show that any of the three equations associated with each node is redundant with respect to the remaining two and Equation (4.1). This fact can be easily verified by noticing that

the state indexes used for the three constraints (4.2), (4.3), (4.4) are a partition of the whole set of indexes, thus their Activity Shares sum up to the left hand-side term of Equation (4.1). In the following formulation of my inference problem, I consider Equation (4.4) redundant for all nodes with respect to Equations (4.1)-(4.3).

I conclude this section with two remarks. First, the report-based constraints, which are key to my inference methodology, are exclusively based on node cumulative temporal channel observations. This makes my methodology robust to heterogeneous packet lengths and physical transmission bit-rates of the participating nodes. Second, the assumption that the links in E are fixed plays a crucial role in enforcing the constraints in Equations (4.3)-(4.4). Even though this is a simplifying assumption, related research shows that threshold-based carrier sensing relationships can be reasonably well approximated as binary [53]. Our experimental results (see Section 4.5.3), and a specific discussion in [45], evaluate the effects of this assumption in a static indoor environment. It is part of my ongoing work the generalization of the report-based constraints to encompass cases of very high channel variability. Specifically, I am considering to associate weights to the Activity Share elements in the left-hand-side summations of Equations (4.3)-(4.4), representing the probability that the signal strength received by node k when the network is in state \vec{D}_j overcomes the carrier sense threshold.

4.3.3 Entropy-based Statistical Solution

In this subsection, I show how to determine a representative \vec{AS} close to the actual \vec{AS} occurred during the measured interval. The representative \vec{AS} should satisfy the report constraints, since the actual \vec{AS} determines the reported measurements. However, the constraints I defined do not identify in general a single \vec{AS} , but rather a feasible solution domain. Each Activity Share distribution \vec{AS} in the domain defined by the reports would have generated the exact same observations obtained by the nodes; hence, the selection of any of these \vec{AS} is admissible. However, a key observation is that not all feasible solutions are equally likely, e.g., 802.11 introduces a bias against states that include simultaneous transmissions of mutually carrier sensing nodes. I formalize this bias using the *a priori* distribution of the states, and I select my representative \vec{AS} as the feasible solution closest to the *a priori* distribution.

Protocol-driven a priori information. As shown in Section 4.2.1, I can give a statistical interpretation of the components of the Activity Share. Each $AS(\vec{D}_j)$ corresponds to the probability the network is in the state \vec{D}_j at a random time instant. Because of the carrier sensing behavior of 802.11, not all network states have *a priori* identical probabilities of occurrence, i.e., $AS(\vec{D}_j)$ is *not a priori uniform* (i.e., equal to $\frac{1}{2^N}$) over all states \vec{D}_j . In fact, 802.11 carrier sense aims to prevent the occurrence of states where neighboring nodes transmit simultaneously, i.e., $\{\vec{D}_j \mid \exists k, l : l \in V_{cs}(k), D_j^k = 1, D_j^l = 1\}$. Practically, two neighbors can transmit

simultaneously only if their backoffs expire in the same slot, while non-adjacent nodes can initiate their transmissions independently. In general, the larger is the number of neighboring transmitting nodes in a state, the lower is the probability of occurrence of that state, since such occurrence would require that a number of backoff counters expired exactly in the same slot. As a consequence, among the admissible \vec{AS} , my scheme should favor the \vec{AS} that do not assign large probabilities to states including neighbor transmissions.

I model the protocol behavior of 802.11 by identifying an *a priori* distribution that assigns probabilities to the states \vec{D}_j unequally. The computation of the exact *a priori* probability of each state is complicated, because the probability of occurrence of states including multiple adjacent transmitters depends on the global network topology. In order to provide a simple solution, I use a coarse-grained approximation that assigns to each network state an *a priori* probability exponentially decreasing with the number of adjacent transmitters the state contains. For example, a state containing two pairs of adjacent transmitters has half the *a priori* probability of a state that contains only one pair. Notice that this assignment partitions the states \vec{D}_j in classes, where all the states in the same class contain identical numbers of adjacent transmitters, and thus have equal probabilities. For instance, class 0 includes all states that do not contain adjacent transmitters and have probability p , class 1 includes all states that contain only one pair of adjacent transmitters and have probability $p/2$,

etc.

Minimum Relative Entropy \vec{AS} inference. In the previous paragraph, I formalized my knowledge of the protocol behavior by using an *a priori* distribution of \vec{AS} . Our objective is to select the feasible \vec{AS} closest to the defined *a priori* distribution. I propose to use the concept of Kullback-Leibler distance [14] to quantify the distance between two distributions, and select the representative \vec{AS} as the feasible solution that minimizes such distance from the *a priori* distribution. Accordingly, the problem is formulated following the Minimum Relative Entropy Principle.* Out of the feasible solutions that have equal Kullback-Leibler distance from the *a priori* distribution, the Minimum Relative Entropy Principle favors the solutions that spread the probability of the states in the same class as evenly as possible. In fact, in absence of any other information about the 802.11 protocol behavior, all states that the *a priori* distribution assigns to the same class have identical probability. Hence, any different probability assignment would introduce an unmotivated bias.

*Note the that minimizing the relative entropy is equivalent to maximizing the expected value of the log-likelihood.

The \vec{AS} Inference problem. I formulate the \vec{AS} inference problem as:

$$\min_{\mathbf{x}} \quad \sum_{j=0}^{\gamma-1} x_j \log \frac{x_j}{w_j} \quad (4.5)$$

$$\text{s.t.} \quad \Phi \cdot \mathbf{x} = \mathbf{T}$$

$$\Psi \cdot \mathbf{x} = \mathbf{B}$$

$$\mathbf{1}' \cdot \mathbf{x} = 1$$

$$\mathbf{x} \geq \mathbf{0}$$

where γ is the cardinality of the set of admissible network states (2^N in this case); \mathbf{x} is a γ -dimensional vector, whose j -th entry, x_j , is $AS(\vec{D}_j)$; \mathbf{w} is a γ -dimensional vector, whose j -th entry, w_j , is the *a priori* distribution of the network state \vec{D}_j ; Φ is an $N \times \gamma$ matrix, whose ij -th entry is 1 if $D_j^i = 1$, 0 otherwise; Ψ is an $N \times \gamma$ matrix, whose ij -th entry is 1 if $D_j^i = 0$ and $\exists s \in V_{cs}(i) : D_j^s = 1$; \mathbf{T} and \mathbf{B} are N -dimensional vectors, whose k -th entries are the measurement results T_k , and B_k respectively. Notice that the objective function is the relative entropy between the solution \mathbf{x} and the prior distribution \mathbf{w} ; further, the first and second constraints (each N -dimensional) correspond to Equations (4.2) and (4.3) respectively, while the third constraint (1-dimensional) corresponds to Equation (4.1).

4.3.4 Protocol-based State Space Reduction

The solution space of the \vec{AS} inference problem is generated by 2^N variables, i.e., the Activity Share components that correspond to all possible network states; as the number of network nodes N increases, the exploration of such a large space to find the best candidate solution becomes computationally complex. In order to reduce space and complexity, I again leverage the protocol properties of 802.11 which permit to discover unlikely states.

As I observed, due to carrier sensing, the occurrence of \vec{AS} that assign large probabilities to states including neighboring transmissions is unlikely. I take advantage of this consideration by excluding from the solution space the \vec{AS} with $AS(\vec{D}_j) > 0$, for any \vec{D}_j including neighboring transmitters. Practically, this is equivalent to reducing the number of Activity Share components, by eliminating those corresponding to the unlikely \vec{D}_j . In terms of graph theory, the set of transmitters in any allowed state is an independent set of the graph G . Thus, the number of network states, and of Activity Share components to be estimated, reduces to the cardinality of the set of the independent sets, which is generally still exponential (in graphs with bounded node degree [17]) but smaller than 2^N .

By using this simplification, the resulting inference problem can be obtained from Problem (4.5), by equating γ to the cardinality of the set of the independent sets of the network and by replacing w_j with $\frac{1}{\gamma}$, $\forall j$. The latter substitution reduces the

Minimum Relative Entropy objective to Maximum Entropy: the probability of all the states in the \vec{AS} solution will be spread as evenly as possible according to the constraints.

In my experiments, I verified that the enhancement described above permits to double the network size that I can solve with similar time budget. While simplifying the computation, the illustrated state space reduction is only an approximation of the reality and may penalize the accuracy of the obtained solution. I investigate the performance of the state space reduction in Section 4.5.4, while I adopt the full state space representation in the testbed results in Section 4.5.3.

4.4 Mitigation of Hinderings Transmissions

In this section, I address my goal of improving the throughput of under-served links. Specifically, I show how MIDAS uses the Activity Share to predict how limiting the transmission rate of any hindering node will benefit the throughput of the problematic link. The manager uses my prediction tool to anticipate the outcome of alternative corrective actions (a corrective action is a pair of conflicting node and rate-limiting amount), and to choose the most profitable according to the management policy. The rate-limiting amount of any conflicting node may be determined, e.g., considering the current throughput surplus of the links that node forms, with respect to a previously agreed minimum. In case rate-limiting a single conflicting node reveals insufficient

to reach the target throughput on the under-served link, the manager iterates the evaluation after collecting new reports.

Our prediction tool is comprised of two procedures: *i)* I address the main challenge of estimating the Activity Share after a potential corrective action; *ii)* based on the new Activity Share, I estimate the potential throughput gain that any single link can obtain, in particular the target link. With regard to the first procedure, the key technique I devise follows a differential approach in which I consider that small deviations from the current network conditions have limited effect on the nodes other than the rate-limited and the under-served. The second procedure uses a simple model that identifies how the Activity Share affects the busy time and collision probability of the under-served link. In this section, I discuss each step separately.

4.4.1 Evolution of the Activity Share after Rate-limiting

In order to obtain the potential throughput gain of the under-served link by rate-limiting a specific node (Section 4.4.2), I first compute the Activity Share after rate-limiting. Our methodology follows a differential approach that assumes that small changes on the transmission rate of a node do not affect the relative durations of the states in which that node transmits. In particular, I assume that the Activity Share of the states in which the rate-limited node transmits will reduce in proportion to their values before rate-limiting. Note that based on the differential approach, the total

time the nodes transmit, other than the under-served and rate-limited nodes, is not affected by the change. In practice, this can be realized, e.g., by having the transmission rates of neighboring links fixed to the value before the management operation. In the following, I illustrate the analytical aspects of the differential approach, while its accuracy is implicitly evaluated by the experimental results in Section 4.5 (see in particular, Figures 4.8 and 4.18-4.20).

Denote AS^o (Activity Share *old*) and AS^n (Activity Share *new*) the Activity Share before and after the rate-limiting action, respectively. Let us consider the case of rate-limiting the packet transmission rate (i.e., at the MAC layer) of a single conflicting node k of a quantity RL_k . I define $\{\vec{D}_l^{k0}\}$ the states in which k does not transmit (i.e., $D_l^k = 0$), and $\{\vec{D}_l^{k1}\}$ the states in which k does (i.e., $D_l^k = 1$), and I establish that the j -th states, i.e., \vec{D}_j^{k0} and \vec{D}_j^{k1} , differ only for the k -th entry, i.e., $\vec{D}_j^{k0} = \{d_{j1} \dots d_{jk-1} 0 d_{jk+1} \dots d_{jN}\}$ and $\vec{D}_j^{k1} = \{d_{j1} \dots d_{jk-1} 1 d_{jk+1} \dots d_{jN}\}$. Using the differential approach, the Activity Share of the network states (in $\{\vec{D}_l^{k1}\}$) in which k transmits decreases proportionally to the duration of those states in AS^o , and the state \vec{D}_j^{k0} benefits from the decrease of the state \vec{D}_j^{k1} , for all j . Formally,

$$AS^n(\vec{D}_j^{k1}) \approx AS^o(\vec{D}_j^{k1}) - \frac{AS^o(\vec{D}_j^{k1})}{\sum_{l:D_l^k=1} AS^o(\vec{D}_l)} \cdot h \cdot RL_k \quad (4.6)$$

$$AS^n(\vec{D}_j^{k0}) \approx AS^o(\vec{D}_j^{k0}) + \frac{AS^o(\vec{D}_j^{k1})}{\sum_{l:D_l^k=1} AS^o(\vec{D}_l)} \cdot h \cdot RL_k \quad (4.7)$$

where h is the duration of the packets sent by k , and RL_k is the rate-limiting amount of node k in terms of packets per second. For ease of exposition, I assume fixed duration of the data packets transmitted over all links; the use of different bit-rates and thus packet durations on different links may be accommodated extending h to a vector form. Next, I will use the AS^n to obtain the new collision probability of the under-served link.

4.4.2 Relationship between the Collision Probability of the Under-Served Link and the Activity Share

According to [20], I can express the maximal throughput of any link after the rate-limiting action by estimating the new busy time of its sender and the new collision probability. The new busy time of the sender can be obtained from the new Activity Share using Equation (4.3). In this section, I show how to use the new Activity Share to determine the new collision probability of any link, and in particular of the under-served. For the sake of simplicity, my derivation only considers the packet collisions with hidden terminals, which are typically the overwhelming majority.

Given the Activity Share, the main challenge in computing the collision probability is in the transformation of the cumulative time the colliding nodes have transmitted simultaneously into the number of collided packets. I illustrate this issue via a simple example. Consider an underserved link (a, b) affected by a hidden terminal c , and suppose that I aim to determine the collision probability on (a, b) using the Activity Share distribution. Let τ be the sum of the Activity Share of the states where nodes a and c transmit simultaneously. Since a packet on (a, b) can collide at most with two different packets sent by c (assuming a fixed and identical duration h of the packets sent by a and c), the total number of packet collisions that c may have caused on (a, b) can be any integer in the range $[\frac{\tau}{h}, 2 \min\{Pkt_a, Pkt_c\}]$, where Pkt_a (resp. Pkt_c) denotes the number of packets transmitted during the observation interval by node a (resp. c). This shows that a large range of collision probabilities is consistent with the same Activity Share distribution. I remark that the example above is only provided for illustrative purpose, and is not intended to limit the applicability of my solution. In the following, I use a binary channel assumption; accordingly, a packet on (i, j) is corrupted if it overlaps for any arbitrary small duration of time with any other packet reception at j .

In order to compute the collision probability $p^{i,j}$ of a problematic link (i, j) , I determine the success probability, i.e., the probability that the transmission of a packet from i to j entirely fits within a time interval during which its hidden terminals

are not transmitting. To estimate this probability, I model the transmission attempts of i as the sampling of an ON/OFF process representing the aggregate transmissions of all the hidden terminals of i [20, 54]. The ON period is the interval during which at least an hidden terminal is transmitting, the OFF period is the gap in the activity of all the hidden terminals that node i has to discover randomly.

In the analysis of this process, I make the following assumptions. 1) In general, the transmissions of the hidden terminals are not coordinated and may overlap; thus, the durations of the ON and OFF periods are variable. In this case, it is a common assumption to model them distributed exponentially. 2) The duration of an ON period can range from very short, e.g., an individual ACK transmission, to much longer than the duration of a data packet h , in case of consecutive overlapping transmissions of different hidden terminals. I balance these cases, by approximating the average duration of an ON period, \bar{T}_{ON} , with h .^{*} 3) Conditioned on the fact that i can transmit, i.e., that the nodes in $V_{cs}(i)$ are not transmitting, I assume that the transmissions of i occur at random points in time.

In order to succeed, a packet transmitted on (i, j) needs to start during an OFF period, and be entirely received during the OFF period. Thus, using assumptions

1) and 3), I can write the collision probability as: $p^{i,j} = 1 - \frac{\bar{T}_{OFF}}{T_{ON} + T_{OFF}} e^{-\frac{h}{T_{OFF}}}$ [20];

^{*}Notice that, in case different links use different bit-rates and thus packet durations, the average duration of an ON period may be approximated as the average of the durations; the evaluation of this enhancement is beyond the scope of this paper.

assumption 2) permits to obtain $p^{i,j}$ as a function of $\frac{\bar{T}_{ON}}{\bar{T}_{ON}+\bar{T}_{OFF}}$. In the remainder, I show how to express $\frac{\bar{T}_{ON}}{\bar{T}_{ON}+\bar{T}_{OFF}}$ (and thus $p^{i,j}$) as a function of the Activity Share.

In order to do this, I compute the total duration the process is in ON and $\{ON \text{ or } OFF\}$ states during a measurement interval ΔT : the ratio between these two quantities is equal to the ratio of their averages $\frac{\bar{T}_{ON}}{\bar{T}_{ON}+\bar{T}_{OFF}}$. Recall that the ON and OFF states model the sampling of node i of the channel at the receiver, and that node i cannot sample the ON/OFF process (i.e., transmit) during the transmissions of nodes in $V_{cs}(i)$. Hence, I prune all time intervals in which at least one of i 's neighbors is transmitting, i.e., I consider only time intervals in which no node in $V_{cs}(i)$ is transmitting. Thus, the whole duration of the ON/OFF process in ΔT is $(1 - B_i)\Delta T$. Let us denote $V_{ht}(i, j)$ the set of hidden terminals of (i, j) . Then, the whole duration of the ON period in ΔT is the time at least one hidden terminal is transmitting and no node in $V_{cs}(i)$ is transmitting. By using the Activity Share, I denote the latter interval as $AS^{HT*}\Delta T$, where

$$AS^{HT*} = \sum_{l: (\exists m \in V_{ht}(i, j): D_l^m = 1) \wedge (D_l^n = 0, \forall n \in V_{cs}(i))} AS(D_l) \quad (4.8)$$

Finally, the identity between $\frac{\bar{T}_{ON}}{\bar{T}_{ON}+\bar{T}_{OFF}}$ and the ratio of their total durations in ΔT discussed above leads to

$$\frac{\bar{T}_{ON}}{\bar{T}_{ON} + \bar{T}_{OFF}} = \frac{AS^{HT*}\Delta T}{(1 - B_i)\Delta T} \equiv AS^{normHT*} \quad (4.9)$$

By replacing (4.9) into $p^{i,j}$, I can write:

$$p^{i,j} = 1 - (1 - AS^{normHT*})e^{-\frac{AS^{normHT*}}{1 - AS^{normHT*}}} \quad (4.10)$$

which expresses the collision probability of a link using exclusively the Activity Share. Using Equation (4.10) I can compute the throughput according to [20].

4.5 Performance Evaluation

In this section, I validate MIDAS through an extensive set of testbed and simulation experiments. After introducing my experimental platform and implementation, I investigate the performance of MIDAS in a real testbed deployment. Finally, I extend the evaluation by simulating a broader set of topologies with larger numbers of nodes, in order to determine the sensitivity of the tool to node density and traffic load, and show its robustness to missing reports and real traffic distribution. Additional results can be found in [44, 45].

4.5.1 Experimental Testbed

WARP. To validate MIDAS, I used the Wireless Open-Access Research Platform (WARP) developed at Rice University [3]. The platform, built around a Xilinx Virtex

processor, includes the MAX2829 radio chipset that provides RSSI readings. Moreover, WARP implements an OFDM layer similar to 802.11a. In my configuration, the boards operate at 6 Mbps using BPSK modulation, and are equipped with a 3 dBi antenna; all boards are controlled by a laptop via Ethernet connections.

Inference Tool Implementation. The implementation of the inference tool consists of two basic components. *i)* The *transmission duration counter* measures the time duration the radio is in transmission state by timing the functions that control the transmission operations. *ii)* The *sub-packet RSSI time sampler* measures the time duration the received signal strength, including noise and interference, exceeds a given threshold. In contrast to existing off-the-shelf drivers, such as MadWifi for Atheros chipsets,* which only provide an RSSI sample per packet, my implementation samples the RSSI values at regular time intervals shorter than the packet duration, and compares them to the carrier sensing threshold.

Validation Tool. Two additional components were implemented only for validation purposes. *i)* The *fast RSSI sampler* behaves identically to the sub-packet RSSI time sampler described above, but supports higher sampling rates via a digital design, thus improving the precision of the busy time estimation. *ii)* The *trace collection logic* provides the ground truth of my experiments by collecting and storing on the board's memory the timestamps and durations of all radio-transmitted packets

*Multiband Atheros Driver for Wifi. Available at <http://madwifi.org/>

and sends batch traces to a control station. The individual node traces are **not** used by the inference tool, but permit to reconstruct offline a network-wide global trace of the transmitting activity of all nodes and to extrapolate the actual Activity Share. In order to synchronize the individual traces from different nodes, the control station issues an Ethernet broadcast to the boards at the beginning of each experiment, which is used to reset their clock. I verified that my technique achieves clock offsets below a few micro-seconds.

Testbed Setup. I conduct my experiments on a five-node indoor testbed. In order to verify the robustness of MIDAS to different node densities, I alternately deployed my nodes in different topological configurations. I list the locations used in my topologies in decreasing order of density, with reference to Figure 4.2: in the single-hop topology $S1$ all nodes are next to each other close to position b ; in the multi-hop topology $M1$ the nodes are located in the positions $\{a, b, c, d, e1\}$; in the multi-hop topology $M2$ the nodes are in positions $\{a, b, c, d, e2\}$. Each board transmits 1000-byte data packets, with constant inter-packet time whose value depends on the experiment. Each experiment run lasts 10 seconds and, where not differently specified, the reported results are cumulative over 10 runs.

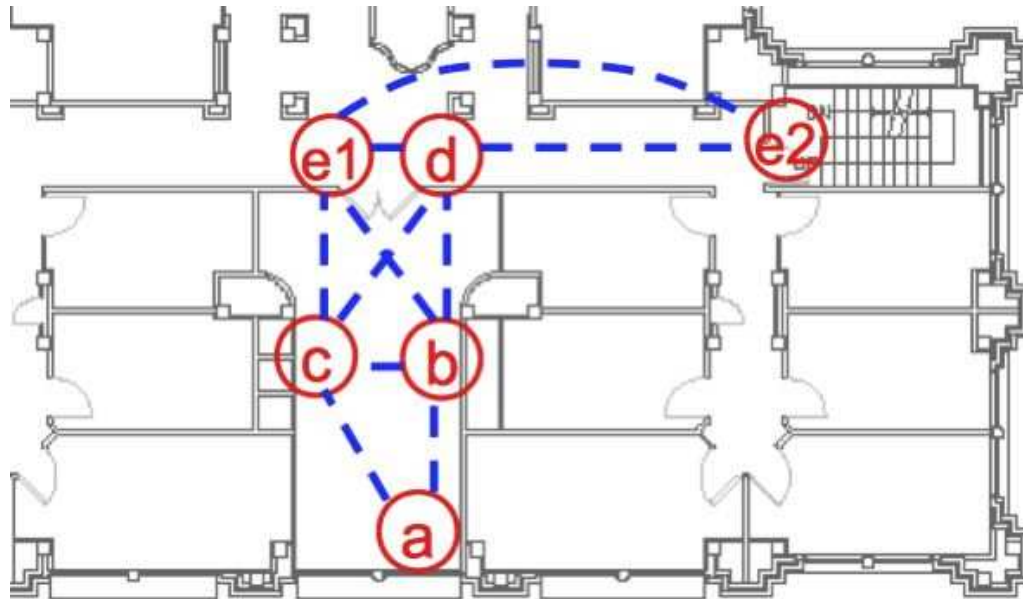


Figure 4.2 Layout of my testbed deployment.

4.5.2 RSSI-based Busy Time Discovery

The challenge in the experiment setup is to devise a technique to consistently measure the node busy time, i.e., the total duration of *all and only* the transmissions of a limited set of neighboring nodes. In Section 4.2.1, I argued that I cannot use packet reception statistics to measure node busy time with the needed accuracy because of packet losses. In this section, I show how to use the received signal strength to discover neighbors' transmissions. The advantage of my technique is that it does not rely on packet decoding, thus being resilient to losses.

I identify the neighbor set of a node as the set of nodes whose transmissions consistently exceed an RSSI threshold. Because wireless links are heterogeneous, the

received signal strength varies from link to link; because of fading, it fluctuates in time. The first issue requires each node to autonomously calibrate its own threshold value, depending on the local topology, so that *all and only* neighbors' transmissions consistently exceed it. The choice of an RSSI threshold sufficiently lower (resp. higher) than the average signal strength of every neighbor node (resp. non neighbor node) permits to cope with the temporal variations.

In order to demonstrate my calibration technique, I design an experiment where I evaluate many possible RSSI thresholds for each node in the topology *M2*. In the experiment, each node in the network takes turns of 10 seconds transmitting broadcast packets at maximum rate, while the others measure the fraction of time the RSSI exceeds a threshold. For each transmitter, I perform the experiment 8 times, increasing the threshold of the potential receivers (i.e., all other nodes) from -90 dBm to -66.2 dBm in steps of 3.4 dBm; for each threshold value, I perform 10 iterations. In Figure 4.3, I show the results I obtained for node *a*; for the other nodes I obtained similar results. The X-axis represents the threshold value, while the Y-axis is the ratio between the total time the received RSSI exceeds the threshold and the total transmission time of the source node. The figure shows that, for certain values of the threshold, node *a* can discover almost all transmissions from a set of nodes and almost none from the others. For instance, for a threshold of -80 dBm node *a* can discover more than 99% of the transmissions of nodes *b* and *c*, and less

than 1% of those of d and $e2$. This threshold defines a neighbor set of node a including b and c . The experiment shows that it is possible to identify suitable RSSI thresholds consistently exceeded by neighbor nodes, and rarely by non-neighbors. In my experiment, the calibration procedure just described was repeated once for each topology at the beginning of each session, and the thresholds were left unchanged during hour-long repetitions. To address unusual situations of severe fading where fixed thresholds cannot be identified, I am investigating a technique based on the evaluation of the threshold crossing probability of each neighbor.

4.5.3 Testbed Results

Experimental Methodology. I evaluate the accuracy of the inference tool, by assessing its predictions in different testbed and simulation settings. At the *end of each experiment* performed, I collect *a single* report from each node including its transmission time and busy time, which represent the parameters \mathbf{T} and \mathbf{B} in Problem (4.5). I compute the optimal solution of Problem (4.5) corresponding to the collected values using the Matlab solver *fmincon*. I establish the accuracy of the Activity Share inference by comparing my estimations with the ground truth provided by an *omniscient centralized approach* based on the collection of detailed traces (see the Validation Tool above).

An example of the results obtained from a single run on topology $M2$ is shown in

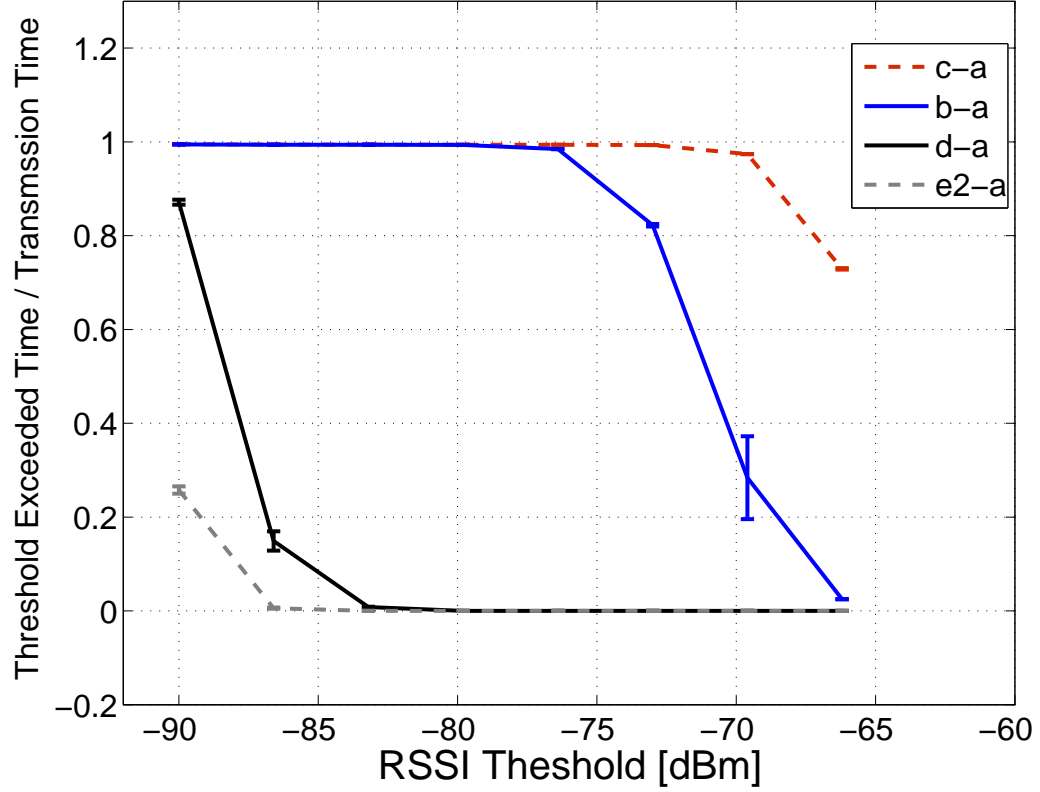


Figure 4.3 Effect of the RSSI theshold on the busy time of node a

Figure 4.4. In the Figure, I present the scatterplot of the predicted and actual (ground truth) Activity Share obtained in the single run. Each value k on the X-axis denotes a *network state* \vec{D} corresponding to the binary representation of k (once mapped the bit indices 0 through 4 to the nodes positioned in a , b , c , d , and $e2$, respectively, e.g., $k = 20$ maps to the network state $\{10100\}$, i.e., where only nodes $e2$ and c transmit). The graph shows an excellent agreement between the inferred Activity Share and the actual Activity Share obtained from the traces. Further, I can observe that a number

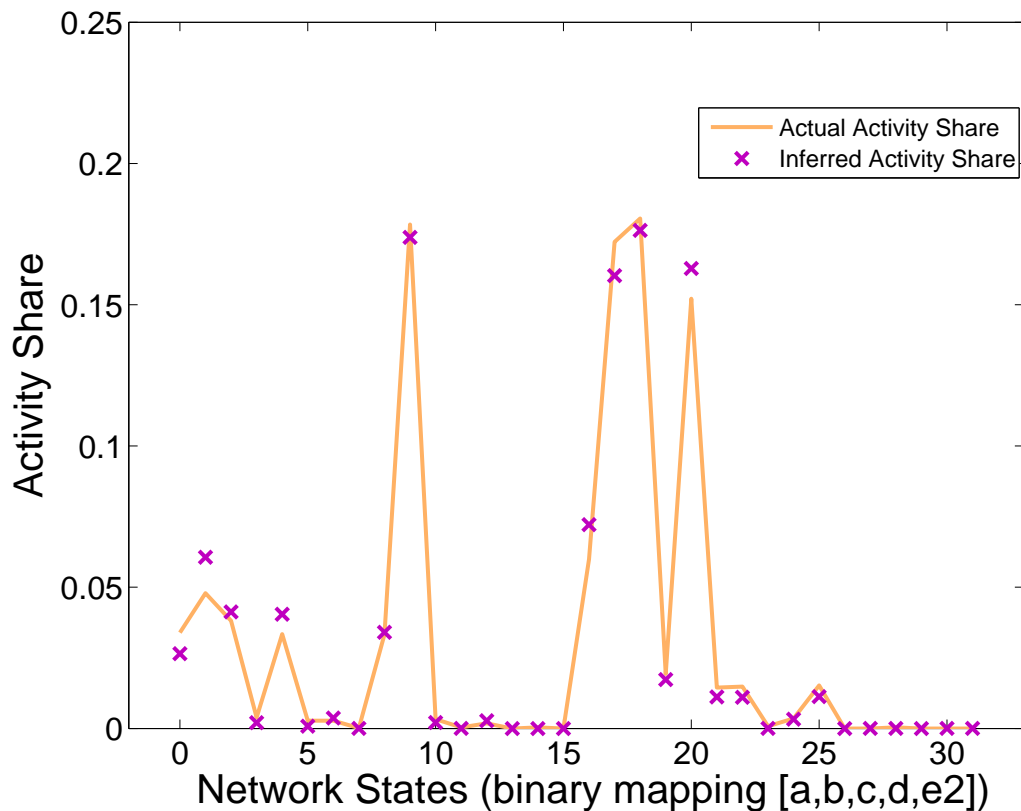


Figure 4.4 Activity Share inference (testbed).

of states have very short durations: these typically include simultaneous transmissions of nodes in carrier sensing range, which occur less frequently than the others.

Sensitivity to Network Density. Network density can highly affect the accuracy of the Activity Share estimation. In order to infer the Activity Share, it is challenging to estimate the duration of overlapping transmissions of non-neighboring nodes, by combining their transmission reports with the busy time share reports of their common neighbors. On the one hand, the lower is the density, the more

tightly the busy time share reports constrain the overlapping transmissions of non-neighboring nodes, but also the fewer reports reflect such events. For example, if a node z has only two transmitting neighbors x and y (where x and y are not neighbors), z 's busy time share report permits to exactly recover the share of time that transmissions of x and y overlapped, as $T_x + T_y - B_z$. However, if a node has three transmitting neighbors (which are non-neighbors to one another), based on the busy time share of the node it is not possible to determine the amount of overlapping transmissions of any two or all three of the neighbors. In fact, the higher the density, the more combinations of overlapping transmissions of a node's neighbors are consistent with the node's busy time share report. On the other hand, the higher is the density, the larger is the number of nodes that observe the transmitting activity of a given set of transmitters. Accordingly, more constraints can be imposed on the Activity Share estimation based on the diversity of the reports of different neighbors. In order to investigate the influence of network density on the Activity Share accuracy, I run experiments on all three different topologies of my testbed: topology $S1$ which is densest, as all nodes are connected to one another, topology $M1$ which is less dense, and topology $M2$ which is the sparsest.

Figure 4.5 shows the CDF of the relative error of the Activity Share inference (notice that the probability of a state used to compute the CDF is the Activity Share of that state, i.e., its duration). The X-axis indicates the relative error committed,

while the Y-axis is in (non-dimensional) time ratio units. For instance, the point in (0.1, 0.7) indicates that the network spends 70% of the time in states where my inference tool commits an error of 10% or less. All plots show that my inference technique is remarkably accurate under all density conditions; further, $S1$ is the most accurate solution, while the $M1$ plot mostly dominates $M2$. The respective mean relative errors, i.e., the relative error committed in the state occupied in a randomly sampled instant, are 4.6% for $S1$, 9.9% for $M1$, and 11.5% for $M2$. These results are obtained for broadcast packets; however, similar values have been obtained using one-hop unicast flows, i.e., 4.8% for $S1$, 6.1% for $M1$, and 7.7% for $M2$. *I conclude that the Activity Share inference tool is accurate under all density conditions: in low density, a small number of reports reflects overlapping transmission events, but they impose tight constraints; in high density, the large report diversity compensates for the looser constraints imposed by the reports.*

The influence of network density on the Activity Share is revisited by simulating larger topologies and the results can be found in [44].

Sensitivity to Traffic Load. Similarly to network density, traffic load can also affect the accuracy of the Activity Share estimation, since it influences the overlapping transmissions sensed by a node. The higher the traffic load, the larger is the amount of overlapping transmissions, which challenge the estimation of the Activity Share by enlarging the feasible state space. In fact, as noted earlier, in case of overlap-

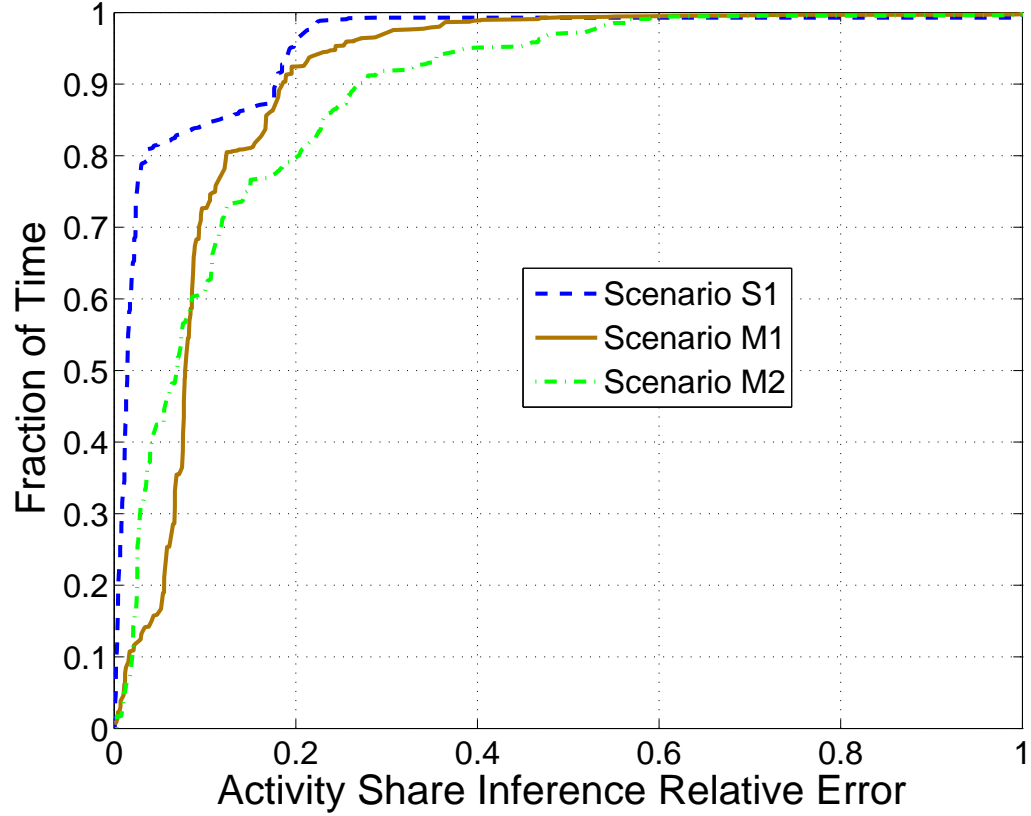


Figure 4.5 Inference sensitivity to network density (testbed).

ping transmissions, several combinations of Activity Shares may generate identical observations (i.e., node busy and transmission time shares). However, light traffic conditions increase the free airtime observed by a node, which in turn weakens the coordination attained by carrier sensing, by decoupling the transmitting patterns of the nodes and leaving larger room to randomness. I study the impact of traffic load on the Activity Share inference tool, by running the experiment on a fixed topology with various traffic loads. Specifically, I iterate scenario *M1* three times, fixing the

traffic loads of all nodes to 400 kbps, 1.2 Mbps, and 2 Mbps (also in this case, each experiment is repeated 10 times).

Figure 4.6 depicts the CDF of the relative error of the Activity Share estimation. As can be seen in the figure, the Activity Share inference tool attains a very low relative error. Furthermore, the variations among the three plots are minimal, and are comparable with the results attained for the fully backlogged case. In particular, the mean relative error is 4.6%, 4.0%, 4.5% for 400 kbps, 1.2 Mbps, and 2 Mbps, respectively. *I conclude that, even though heavier traffic challenges the Activity Share inference by increasing the amount of overlapping transmissions, while lighter traffic increases randomness, the accuracy of my solution is largely independent of the traffic load of the nodes.*

I defer the investigation of the sensitivity of the inference tool to non-uniform traffic patterns over larger topologies, to the simulation section (see the paragraph “Robustness to Real Traffic Distribution” in Section 4.5.4).

Sensitivity to Report Interval Length. In the previous experiments, I used report intervals of 10 seconds, i.e., each node k sent one report every 10 seconds including the busy and transmission time shares B_k and T_k that k measured during the same interval. The report interval introduces tradeoffs of reporting overhead (favoring long intervals), responsiveness to network changes (favoring short intervals), and obtaining statistically significant data (favoring long intervals). In order to clarify

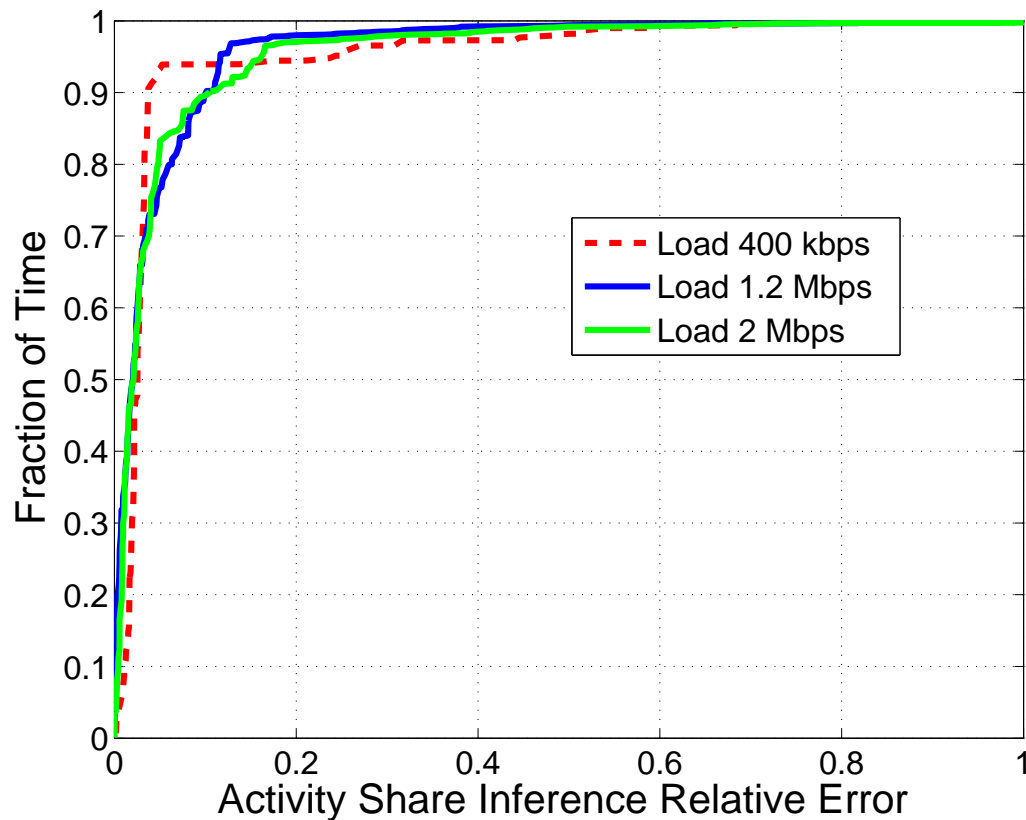


Figure 4.6 Inference sensitivity to traffic load (testbed).

the last issue, I notice that my entropy-based inference tool is most accurate if the node reports reflect steady state observations of the underlying process, i.e., formed by the transmissions of all network nodes (see Section 4.3). As the report intervals shorten, the actual realizations of the process during each interval may largely depart from the steady state, thus degrading the inference accuracy. I assess how short report intervals affect the performance of the inference tool, by measuring the accuracy in the scenario *M1*, for various report interval lengths, varying from 20 s to as low as

100 ms, for fully backlogged traffic.

The experiments show that the inference tool is accurate also for short report intervals (Figure 4.7). In particular, as the report interval decreases from 20 s to 500 ms, the accuracy decrease is minimal. When the report interval is further reduced, and is set up to (as small as) 100 ms, i.e., the reported values are based on approximately 20 packets sent by each node, the accuracy declines. The mean errors are 4.1%, 7.6%, 10.2%, and 29% for the cases of 20 s, 2 s, 500 ms, and 100 ms, respectively. *I conclude that, in order to better capture the network dynamics, the network manager can adapt the duration of the report intervals, with a small penalty on inference accuracy.* Note that since each report includes only two entries, the overhead is minimal. For example, in my implementation, the reports R_k include only two floating point values for a total of 16 bytes, i.e., they easily fit within a single packet, and can be aggregated or even piggybacked in regular traffic.

Throughput Prediction Accuracy With Heterogeneous Concurrent Load.

I evaluate the accuracy of the model in Section 4.4, by comparing its predictions with testbed experiments in the topology $M1$ with single-hop flows $\{a \rightarrow c; b \rightarrow a; c \rightarrow a; d \rightarrow b; e1 \rightarrow c\}$. For each set of experiments, I consider a target under-served link whose traffic is fully backlogged, and I perform a baseline run, measuring the throughput of the target link when all others transmit at rate randomly chosen in the [400 kbps, 900 kbps] interval. At the end of the baseline run, I collect the node

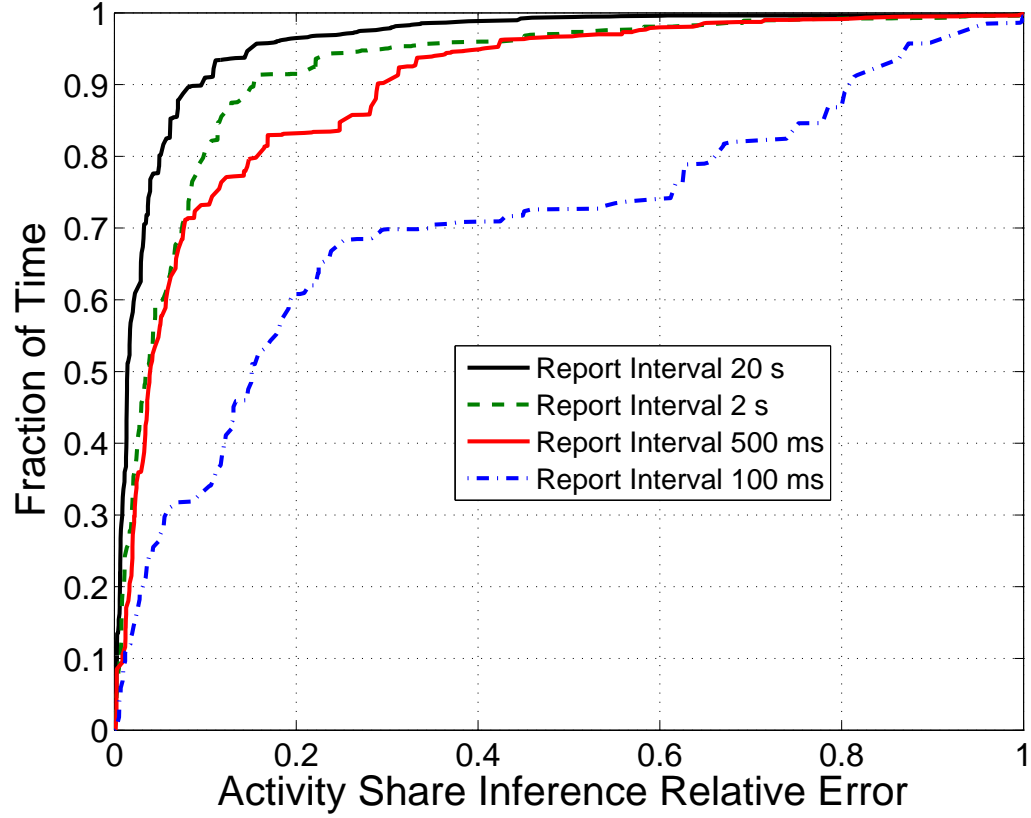


Figure 4.7 Inference sensitivity to short report intervals (testbed).

reports, infer the Activity Share, and predict the throughput increase of the target link obtained by rate-limiting any of the m conflicting nodes of a fixed quantity (400 kbps). Then, I perform m additional runs on the testbed (one per conflicting node), alternately rate-limiting a different conflicting node for the same 400 kbps quantity, and I record the actual throughput gain of the target link. Finally, I compare the actual throughput gain obtained in the testbed with the throughput gain predicted by my model.

Figure 4.8 shows the CDF of the relative error for all possible target link/conflicting node pairs for 10 repetitions of my scenario (200 predictions in total). The long tail of the distribution is due to few combinations for which the actual gain is very small (on the order of a few kbps); in those cases, even an error of few packets is decisive in relative terms. In terms of the absolute error, the predicted throughput gain is on average less than 73 kbps different from the actual throughput gain (i.e., about 18% of the rate-limiting value of 400 kbps, or 26% of the average actual throughput gain of approximately 280 kbps), with the per-link error being {84, 67, 80, 58, 74} kbps (sorted in the alphabetical order of the sender). *I conclude that, despite rate-limiting different conflicting nodes can have largely different impacts on the throughput of an underserved link, my prediction tool adequately captures the heterogeneous effects.*

4.5.4 Simulation Results - Inference Tool

In order to evaluate the inference tool on various topologies including a larger number of nodes, I performed an extensive set of ns-2 simulations following the inference experimental methodology adopted in the previous section. In this section, I first compare testbed and simulation. Then, I evaluate the accuracy loss due to the state space reduction discussed in Section 4.3.4; all the results in the remainder of the paper implement such enhancement. I also investigate the robustness of the inference technique to realistic traffic conditions, and report losses.

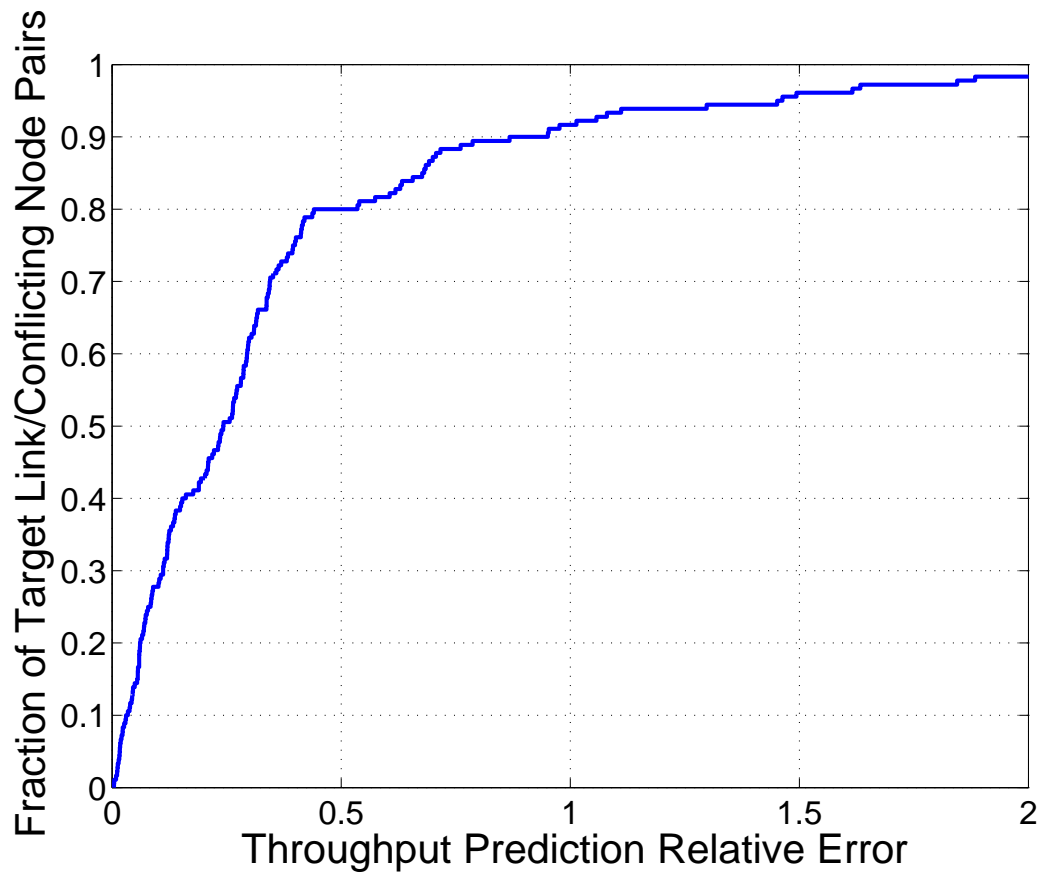


Figure 4.8 Throughput increase estimation for concurrent nodes with loads in [400 kbps, 900 kbps] (testbed).

Simulation Settings. I consider scenarios where each node generates 1000-byte UDP packets directed toward a single neighbor, with constant inter-packet time. The traffic is generated for 100 s at a fixed rate. I use the FreeSpace propagation model, with node transmission and interference ranges equal to 210 m. I generate scenarios with a certain network density (i.e., where each node has on average a predetermined number of neighbors), by deploying the nodes in random positions, and scaling the size

of the deployment area. Except for the experiment in Figure 4.9, which is obtained using 802.11a at 6 Mbps, all results in this section are obtained using 802.11b at 11 Mbps data rate in order to experiment with different conditions. I refer to the analogous simulation results obtained for 802.11a at 6 Mbps as needed.

Comparison between Testbed and Simulations. The simulations introduce simplifications about actual channel propagation and abstract operational details, such as the WARP board’s packet processing time. For this reasons, my first experiment compares the simulations and testbed results. I consider the topologies $S1$ and $M2$ used in the testbed section and fully backlogged nodes. Using the *omniscient centralized approach*, I extract the Activity Share from the traces of simulation and testbed, and I compare them. Figure 4.9 shows the actual Activity Share (Y-axis) for all 32 possible states (X-axis) sorted similarly to Figure 4.4.* *The plots show an excellent agreement between the two environments; the small discrepancies are due to non-ideal packet processing times and carrier sensing relationships in the testbed.*

Effect of the Protocol-based State Space Reduction. The next experiment evaluates the effect of the protocol-based reduction discussed in Section 4.3.4. As remarked therein, the reduction of the state space improves the computational performance of my inference tool by decreasing the size of the feasible solution domain. However, the exclusion from the domain of the states including neighboring transmis-

*Note that for scenario $S1$ the actual node mapping is immaterial.

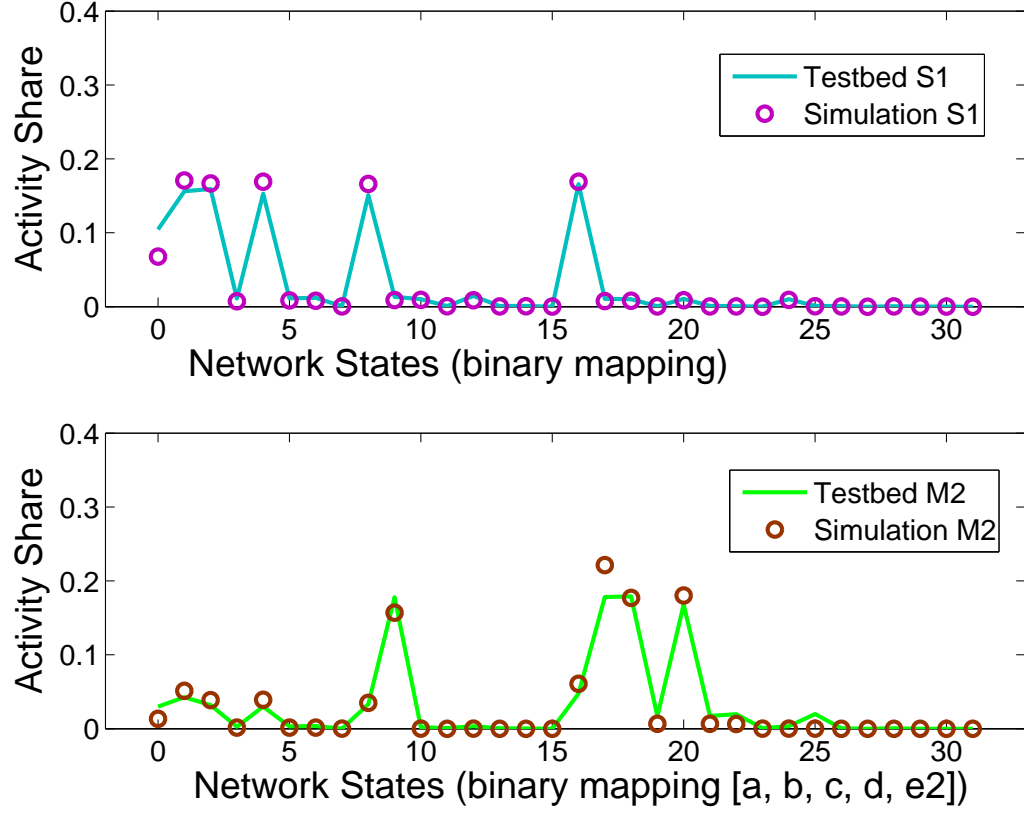


Figure 4.9 Activity Share: simulation vs testbed.

sions introduces an inconsistency between the domain and the measurement reports. In fact, the excluded states likely occurred, even if for relatively short durations, during each report interval and thus influence the reports input to the inference tool. However, the protocol-based state space reduction ignores those states in the construction of the feasible solution domain. In order to assess how this approximation affects the accuracy of the inference tool, I generate a random topology of 10 nodes, with an average number of 7 neighbors per node, and I compare the Activity Share

obtained using the reduced (labeled “Protocol-based Reduction”) and the entire 2^N state spaces (labeled “Power Set”).

Figure 4.10 shows the scatterplot of the Activity Share. The X-axis is the actual value of the Activity Share, while the Y-axis is the estimated value; each mark represents a single state. As expected, the solution including the power set is more accurate (crosses are closer to the line than circles). The concentration of circles on the X-axis close to the origin are due to the states including adjacent nodes transmitting, that the protocol-based reduction excludes. Note that the actual Activity Share values of those states are not significantly larger than 0, as the simultaneous transmissions of neighboring nodes are relatively unlikely. The power set solution benefits from accounting for the unlikely states, not only in the prediction of the Activity Share of those states, but also of states including only independent sets of transmitters. *I conclude that the accuracy of the inference tool is higher when considering the entire state space, since the solution domain of the protocol-based state reduction ignores states that contributed to the reported measurements.*

Sensitivity to Network Density. This subsection revisits the issue of network density on large topologies. In contrast to Section 4.5.3, I run my scheme on the reduced state space. I evaluate the normalized relative error between inferred and actual Activity Share, for 30 topologies of 10 and 15 nodes, with average node neighbors from 3 to 13 and fully backlogged traffic. Recall that the normalized relative

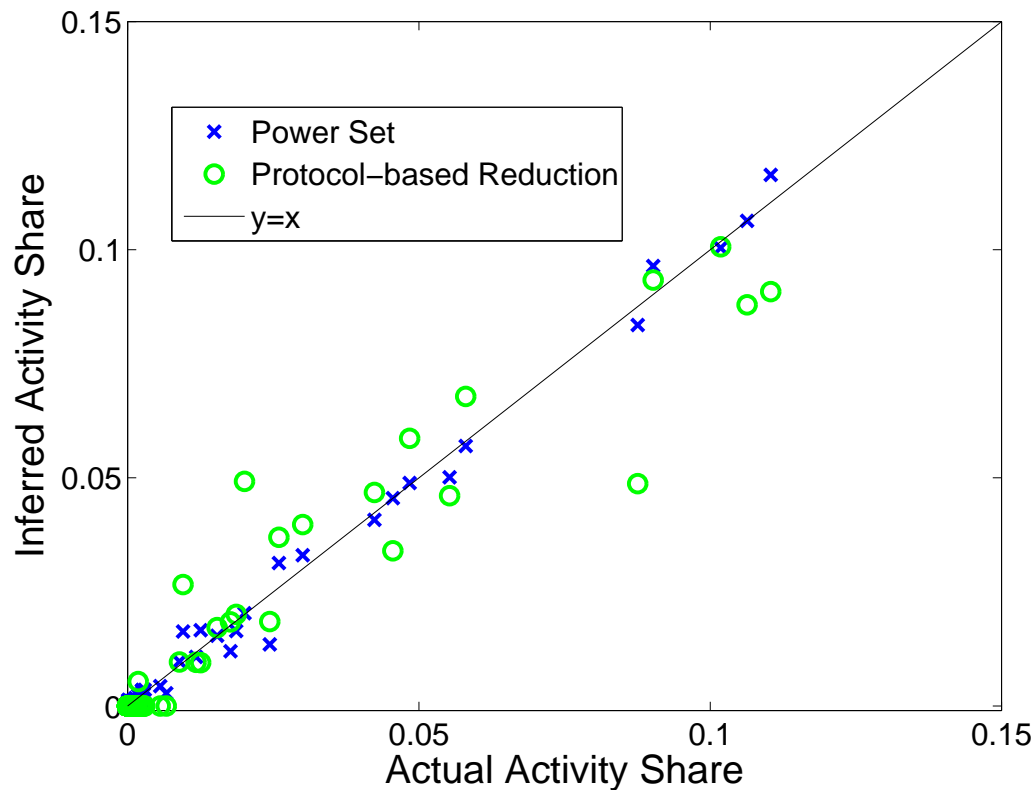


Figure 4.10 Inference with protocol-based reduction.

error is the relative error committed in a state weighted by the Activity Share (i.e., proportionally to the duration) of the state. Figures 4.11 and 4.12 show that my inference tool is accurate under different densities, e.g., the network spends more than 70% of time in states whose relative error is below 20% for all tested densities in 10-node topologies. Figure 4.11 shows that for 10 nodes a density increase from 3 to 5 improves the accuracy of the inference tool, while for density 7 the performance decreases. The average normalized relative errors are 12.2%, 10.2%, 17% for densities

3, 5, and 7 respectively. I ran this experiment also using 802.11a at 6 Mbps, and I obtained normalized relative errors of 13.7%, 12.5%, 15.2%, respectively. Figure 4.12 shows a similar trend for 15-node networks; the accuracy grows for density increase from 3 to 7, but it reduces for density 13. The average normalized relative errors are 18%, 14%, 26%, for densities 3, 7, and 13 respectively. Both figures clearly depict the existence of an accuracy tradeoff related to the network density. As explained in Section 4.5.3, the denser the network the more constraints can be imposed on the Activity Share estimation. However, as the network approaches a clique, the probability of simultaneous transmissions of neighboring nodes increases, thus generating network states that are excluded by the protocol-based state space reduction. For example, the accuracy degrades for 10-node networks with density 7, and for 15-node with density 13. In contrast, in 15-node networks with density 7, nodes in close proximity likely observe different channel busy intervals, due to the diverse sets of carrier sensed nodes; thus, their simultaneous transmissions are less frequent. Notice that, as shown later in more detail, traffic intensity has a similar effect on the validity of the protocol-based approximation, and that fully backlogged traffic is a worst case due to higher occurrence of adjacent node transmissions. *I conclude that, although the protocol-based reduction is accurate in all evaluated settings, high network densities challenge the validity of its approximation.*

Sensitivity to Traffic Load. Not only the network density affects the occurrence

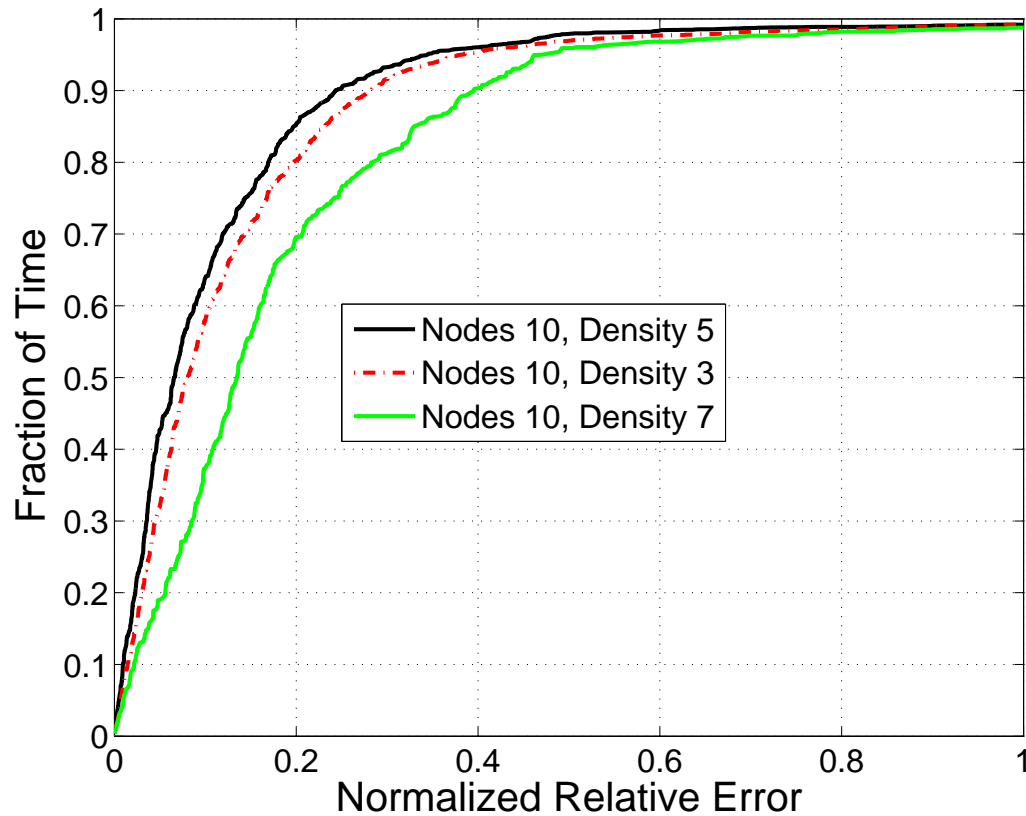


Figure 4.11 Inference sensitivity to density (10 Nodes).

of states excluded by the protocol-based approximation, but also the traffic intensity. In fact, as the traffic decreases, the probability that carrier sensing nodes transmit simultaneously decreases. Figure 4.13 shows the bar graph of the average normalized relative error between inferred and actual Activity Share, for 10 nodes, density from 3 to 7, and target transmission rate from 300 kbps to fully backlogged for each node (in a single scenario, all nodes are subject to identical target transmission rates). The graph shows that the error increases with the target transmission rate for each

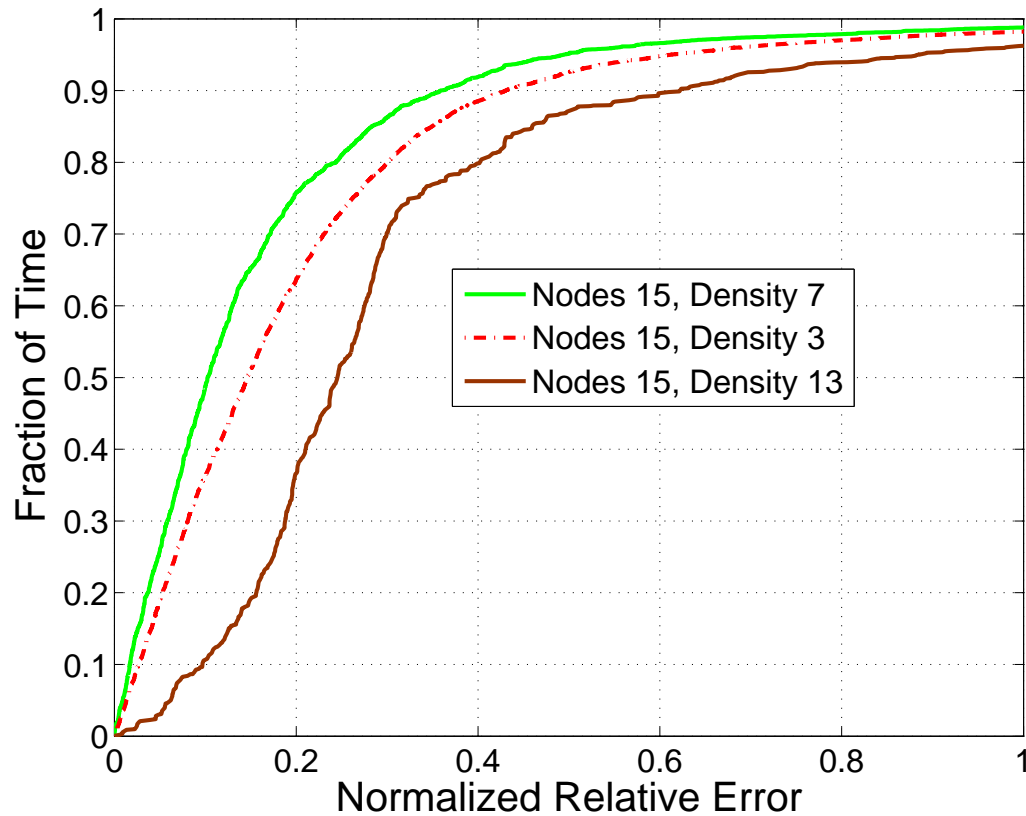


Figure 4.12 Inference sensitivity to density (15 Nodes).

density value. Furthermore, the error increases with the network density, because of the reasons explained above. In particular, as the density increases the network becomes fully backlogged for lower transmission rates (the effect is evident in the case of 900 kbps for density 7). *I conclude that as the network traffic decreases the performance of my methodology improves, because of the increased accuracy of the protocol-based approximation.*

Robustness to Incomplete Information. In the case of severe network conges-

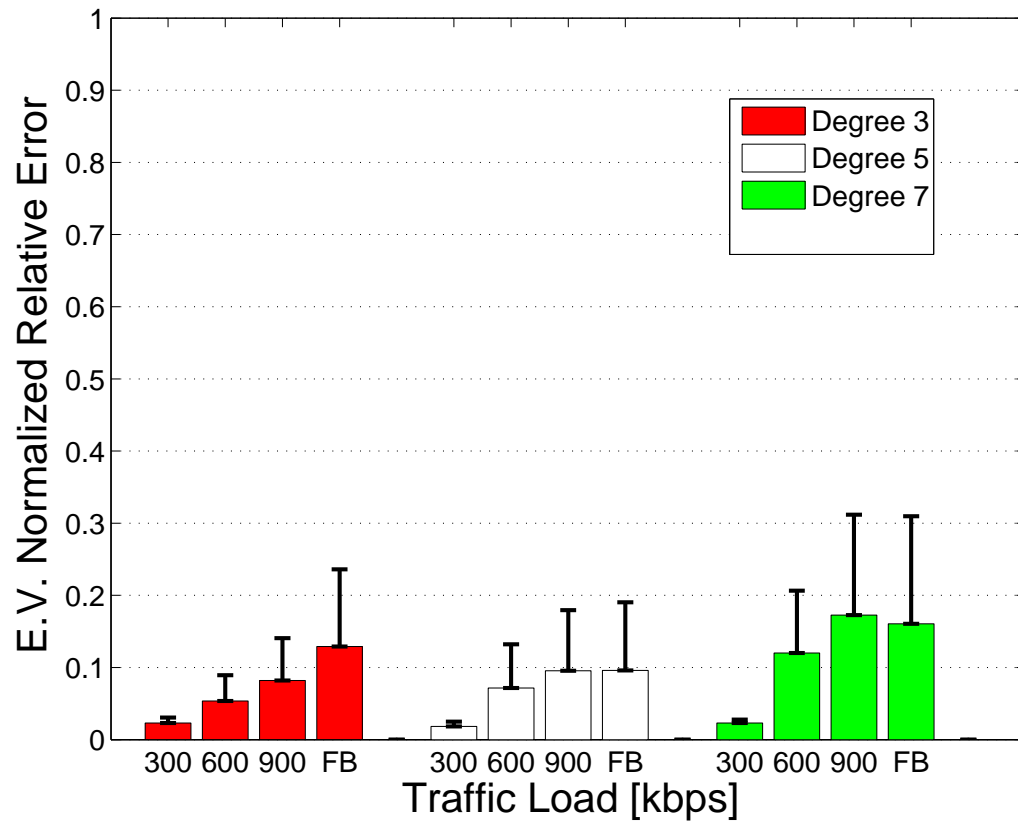


Figure 4.13 Sensitivity of the Activity Share estimation to traffic load (10 Nodes).

tion, some of the reports could be lost. As the number of report losses increases, the accuracy of the inference tool is expected to decrease because of the less constrained, and thus larger, feasible solution domain including the actual Activity Share. A larger solution domain entails a higher uncertainty in the search of the actual Activity Share, since all solutions within the domain may have occurred (see Section 4.3.3). I evaluate how report losses affect the accuracy of the inference tool, by simulating the loss of up to five out of the ten reports transmitted in 10-node networks, with densities of 3, 5 and 7 (i.e., where each node has on average 3, 5, and 7 neighbors, respectively). Figure 4.14 shows the mean relative error of the Activity Share inference computed out of all possible states obtained from 30 random topologies, where I evaluate the lack of all possible combinations of missing reports (bars indicate 85-th percentiles). I observe that the performance gracefully degrades as the number of missing reports increases. This is because the reports of neighboring nodes are related: for instance, part of the busy time of neighboring nodes is generated by transmissions of common neighbors. *I conclude that my inference technique is robust to report losses, due to inherent redundancy of node reports.*

Robustness to Real Traffic Distribution. In my previous experiments, all traffic sources generate packets according to predefined inter-packet distributions. In this experiment, I investigate how critical this assumption may be for my inference technique to operate in real traffic scenarios. In order to reproduce real traffic, I re-

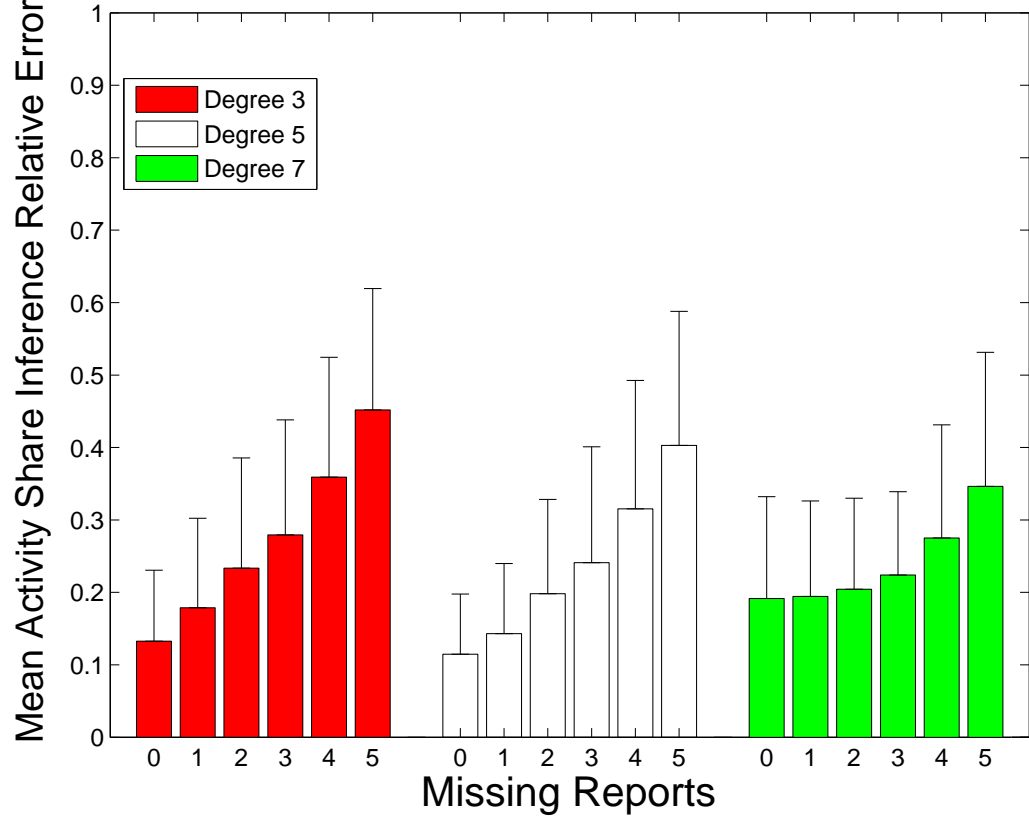


Figure 4.14 Inference robustness to missing reports.

play actual traffic traces collected within UCSD Jigsaw project [12] in my simulation environment. In particular, I randomly select 10 nodes from the UCSD traces, and I play 10 seconds of their traffic on a network topology obtained as follows. Two nodes are considered disconnected if the replayed traces include five overlapping transmissions of data packets from the two nodes (in order to safely consider synchronization errors and simultaneous neighbor transmission events, I consider an overlapping valid only if its duration exceeds $100 \mu s$); otherwise, the two nodes are considered con-

nected. Each simulated node generates packets according to the transmission times of the UCSD node it represents in the trace. Figure 4.15 shows a scatterplot of the measured Activity Share vs. the inferred Activity Share for 10 repetitions of the experiment, with traffic from 10 different time intervals. Note that, in order to visually capture a large range of Activity Share values, I plot both axes in logarithmic scale; for this reason, for small values of the Activity Share the error is visually magnified, although it is only a few percent. The plot shows that also in these real traffic conditions, my inference technique achieves very accurate results. Quantitatively, the mean relative error is about 3%. *I conclude that my inference technique is robust to real traffic conditions, i.e., it is accurate also in case the traffic is not generated according to a predefined distribution.*

Report Interval Length. In this section, I extend the result obtained in the testbed experiments to larger topologies. Furthermore, differently from the testbed experiment, I use the state space reduction which, as shown above, may affect the accuracy of my solution. Specifically, in the previous simulations, I used report intervals of 100 s, i.e., each node k sent one report every 100 s including the busy and transmission timeshares B_k and T_k that k measured during the same interval. This result assesses the accuracy of my inference technique in a 10-node network, with density 5, for report interval lengths as low as 50 ms. Figure 4.16 shows that the inference tool is accurate also for short report intervals. In particular, as the report interval

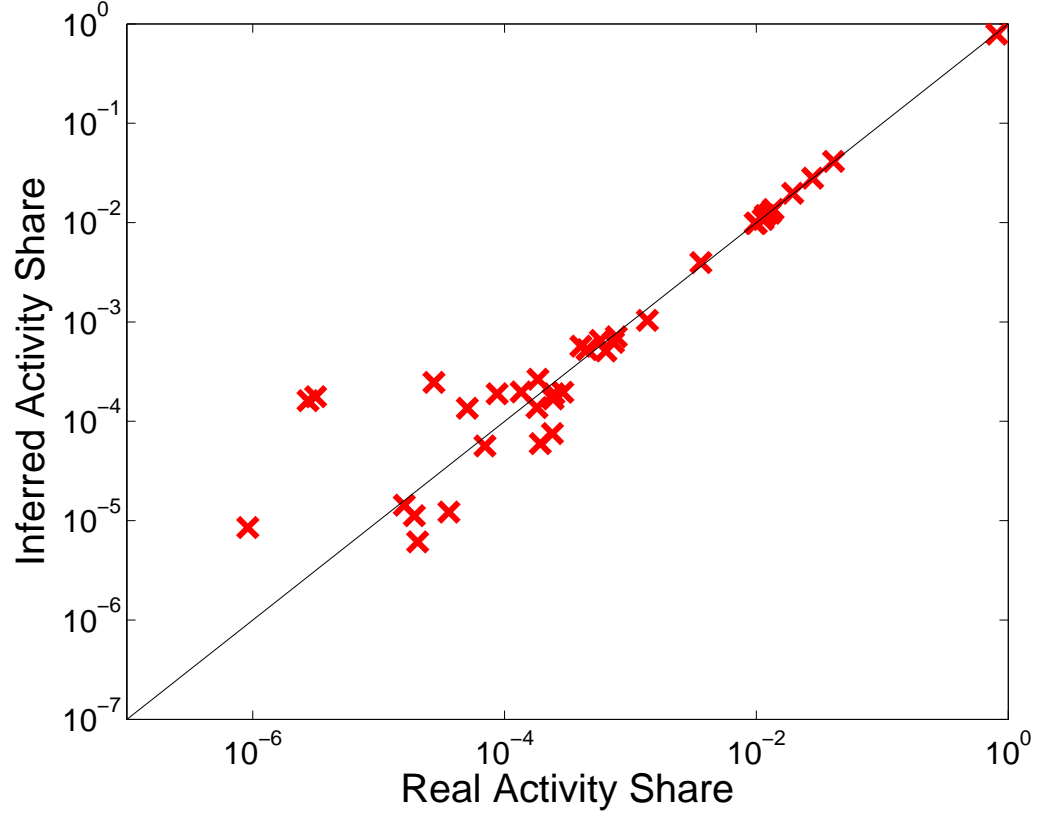


Figure 4.15 Inference robustness to realistic traffic.

is decreased from 100 s to 2 s, the accuracy decrease is minimal. When the report interval is small and set to 50 ms, i.e., the reported values are based on approximately 10 packets sent by each node, the accuracy decreases. The average normalized errors are 10%, 11%, 14%, and 25% for the cases of 100 s, 2 s, 500 ms, and 50 ms, respectively. *Our simulations confirm the conclusions I obtained in my testbed results; the slight performance decrease is mainly due to the state space reduction and to the more complex interactions of larger topologies.*

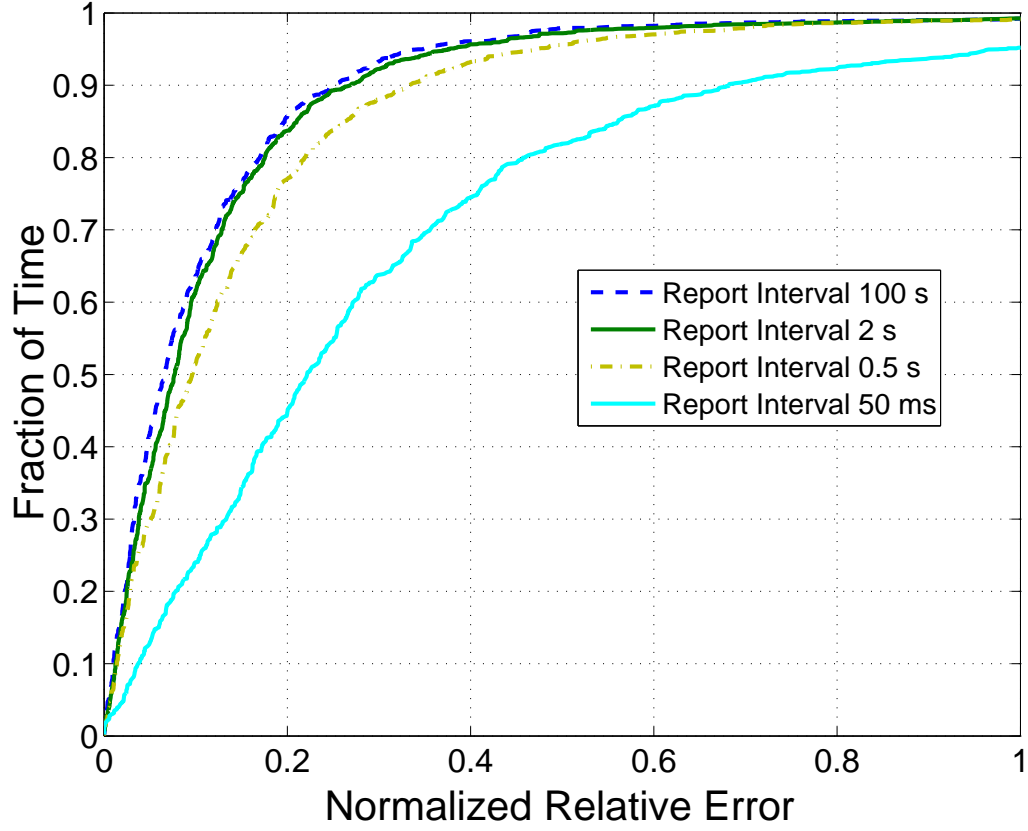


Figure 4.16 Inference robustness to short report intervals.

Comparison with an Exponential Inference Technique. In order to provide a comparison of my inference technique with an alternative base-line method, I design an Exponential Inference Technique based on the assumption that the time duration the system remains in any *network state* is exponentially distributed. This permits us to model the temporal evolution of the system, by using a continuous time Markov chain [19]; notice that this modeling technique is common to several related works [8, 20, 54]. From any state D , the system transitions to the state $D \cup i$ with rate λ_i ,

where λ_i is the transmission rate of i in state D ; the system transitions to the state $D-i$ with rate μ_i , where μ_i is the termination rate of i in state D . Specifically, for the states D for which $D \cup i$ is an independent set, λ_i is equal to the transmission rate of i , once normalized over the duration of the states in which i can transmit because it senses the medium idle (i.e., $1 - B_i$), $\lambda_i = 0$ for the other states; for the states D for which $i \in D$, μ_i is the reciprocal of the duration of a data packet transmission, $\mu_i = 0$ for the other states. For fairness of comparison, differently from all related works, I directly measure the input parameters from the operational network.* I compare MIDAS to the exponential inference solution for topologies of 10 nodes, where each node has on average 5 neighbors, and transmits fully backlogged traffic. Figure 4.17 shows the scatterplot of all the predictions obtained for 10 scenario repetitions, where X's denote MIDAS predictions, while O's denote the exponential modeling solution. The figure shows that MIDAS largely outperforms the alternative, and the mean relative error is 9% versus 36%. *I conclude that a simplifying exponential assumption does not adequately characterize the real system behavior and does not permit to design a technique to accurately infer the Activity Share.*

4.5.5 Simulation Results - Throughput Prediction Tool

I investigate the performance of the prediction tool with ns-2 simulations with the same experimental methodology used to evaluate the throughput prediction accuracy

*Of course, I do not use such inputs in MIDAS (see Section 4.2.2).

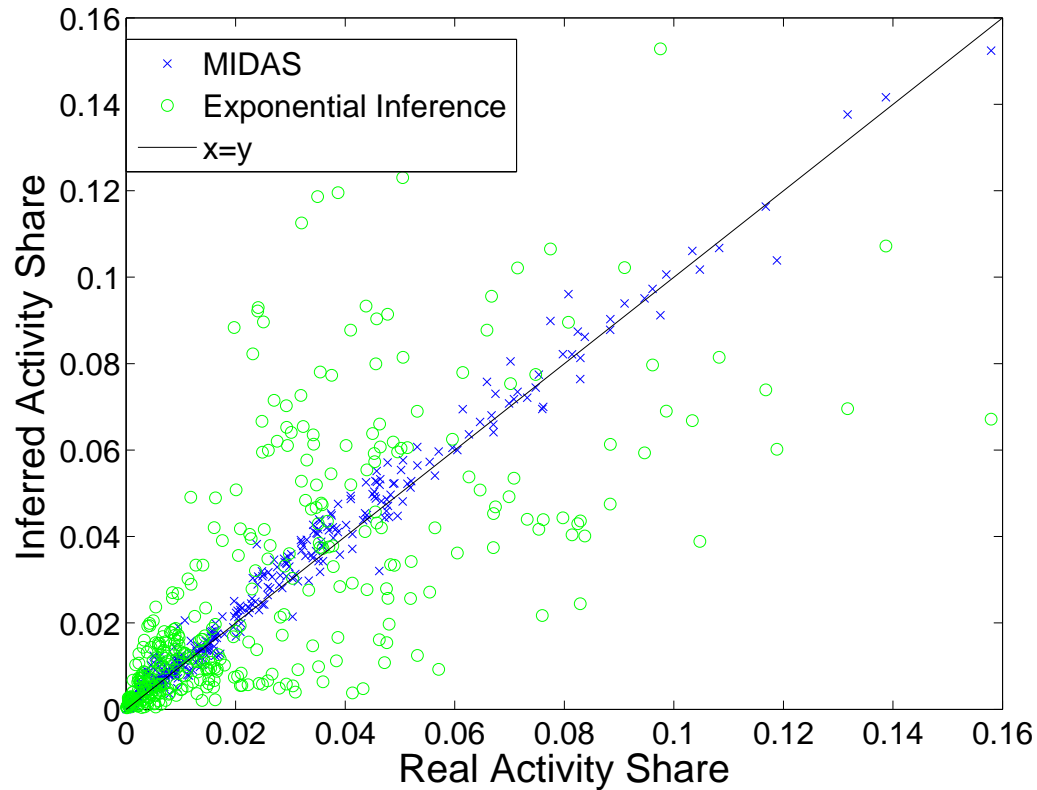


Figure 4.17 Comparison with an Exponential Inference Method

in Section 4.5.3. I start by running MIDAS on a ten node random topology with density 3. Node transmission rates are set to 600 kbps. As in the experimental case, I pick one target underserved link, I increase its load until it is fully backlogged, and successively repeat the experiment, rate limiting each time a different conflicting flow by 400 kbps; I iterate this procedure for all links in the network. The scatterplot in Figure 4.18 compares the predicted throughput gain with the actual throughput increase collected for 10 different random topologies. The X-axis index identifies the

actual throughput increase for a saturated link by rate-limiting one of its conflicting nodes, while the Y-axis represents the predicted value for the same rate-limiting action, e.g., a point on the diagonal represents a perfect match between the actual and the predicted throughput gain of the tagged link; points above and below the diagonal represent an overestimate and an underestimate of the predicted over the actual throughput gain of the tagged link, respectively. The graph shows an excellent agreement between the prediction and the simulation. It is a notable finding that by rate-limiting different conflicting nodes of the same fixed amount, the throughput increase of the target link can range from 7% to 172% of the rate-limited quantity 400 kbps, i.e., from 28 kbps to 688 kbps. In the remainder of this section, I evaluate how the prediction accuracy depends on network density and traffic load.

Sensitivity to Network Density. As previously shown, network density influences the accuracy of the Activity Share inference, which is the basis of throughput prediction (see Section 4.5.3, and [44, 45]). In addition, network density determines the number of neighbors and neighbor’s neighbors (potentially hidden terminals) that respectively affect the link busy time and collision probability, which in turn are key to my prediction tool. I investigate these effects by evaluating my predictions for all possible target link/conflicting node pairs in 10 topologies with 10 nodes, and densities of 3, 5, and 7, with node transmission rates of 600 kbps. Figure 4.19 shows the empirical CDF of the relative error between the predicted and actual throughput

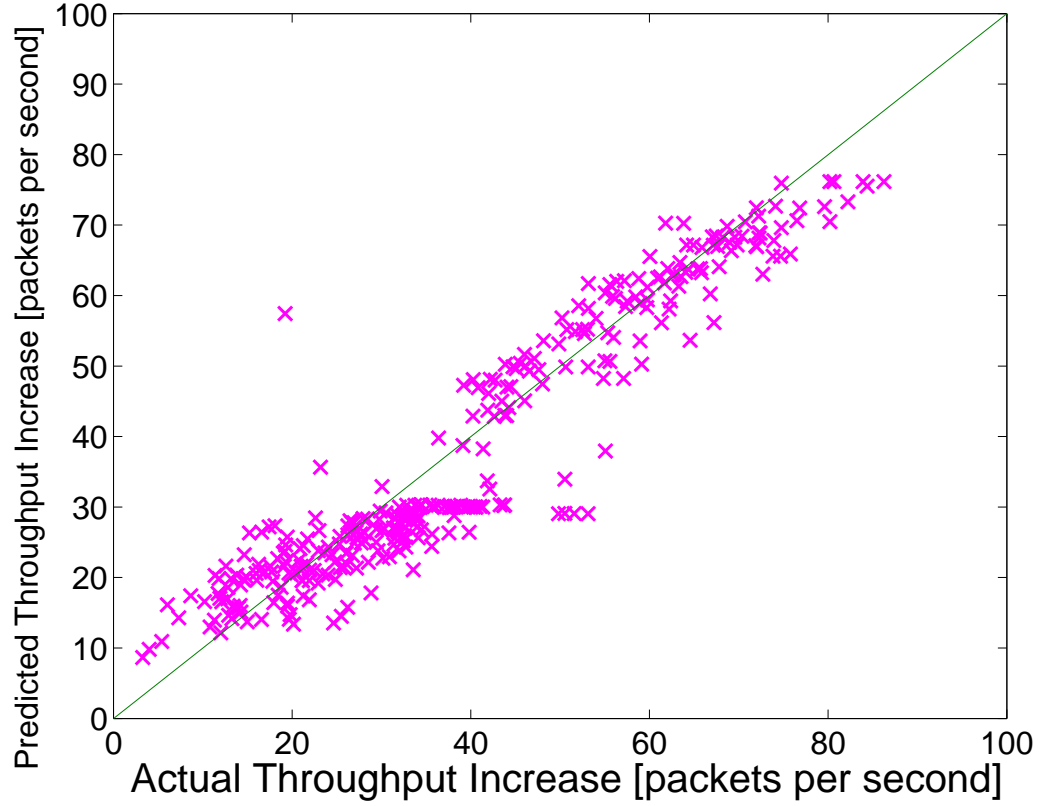


Figure 4.18 Throughput increase estimation for scenarios with density 3.

increase. The plot for density 3 (i.e., for topologies with 3 neighbors per node) is the most accurate, while the case for density 5 is the least; the average relative errors are 17%, 26%, 22% for densities 3, 5, and 7, respectively. Surprisingly, the accuracy in throughput prediction does not exactly reflect the accuracy in the inference of the Activity Share (I checked that the trends in Figure 8 in [44] were respected also in this set of scenarios). The main reason is that my model is more accurate in the computation of the fraction of busy time than of the collision probability, since the former imposes less stringent assumptions (see Section 4.4.2), e.g., my model ignores colli-

sions with terminals in carrier sensing range, whose incidence grows with the density of the scenarios. Thus, the case of density 3, where the number of hidden terminals is restricted by the degree of the receiver, is most accurate. In terms of the absolute error, i.e., the difference between the actual throughput gain and the predicted gain, the predicted throughput gain is within 80 kbps (i.e., 20% of the rate-limiting value of 400 kbps) from the actual throughput gain in 83% to 92% of the cases. *I conclude that the accuracy of the prediction model increases as the number of hidden terminals decreases, because of the less stringent assumptions I impose on the computation of the fraction of busy time of the under-served link.*

Sensitivity to Traffic Load. In this experiment, I investigate the effect of traffic load on the accuracy of my predictions, by repeating the simulations above for node transmission rates of 900 kbps. Figure 4.20 shows the same ranking among the curves relative to different densities as for the case of 600 kbps. However, the accuracy obtained for 600 kbps is higher than for 900 kbps. This is due to two reasons: first, the Activity Share inference technique based on the protocol state-space reduction is more accurate for lower traffic loads (see [45]); second, in terms of the relative error the prediction of small throughput gains is more challenging than the prediction of large gains. As the neighbor load increases, rate-limiting actions produce on average a lower benefit for the under-served link, thus increasing the influence of the less accurate results for lower gains on the CDF. For example, for density 5 and 600 kbps,

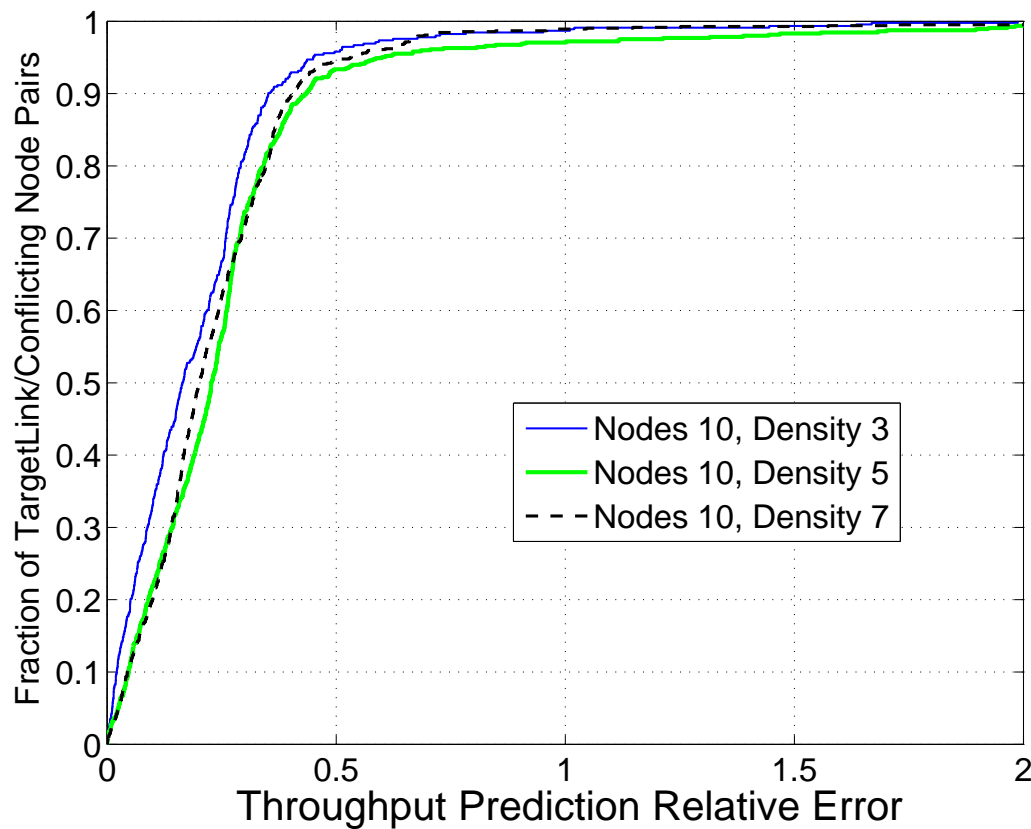


Figure 4.19 Throughput prediction sensitivity to density (600 kbps).

on average the under-served link gains 60% of the rate-limiting amount (i.e., 240 kbps out of 400 kbps in this experiment), while for the case of density 5 and 900 kbps the under-served link gains 40% (i.e., 160 kbps out of 400 kbps). This explains why the relative error is larger for 900 kbps than for 600 kbps.

4.5.6 Simulation Results - Real Network Topology

In this experiment I consider the topology of a real mesh network, TFA [9], deployed in south-east Houston. The topology consists of 18 backhaul nodes whose actual

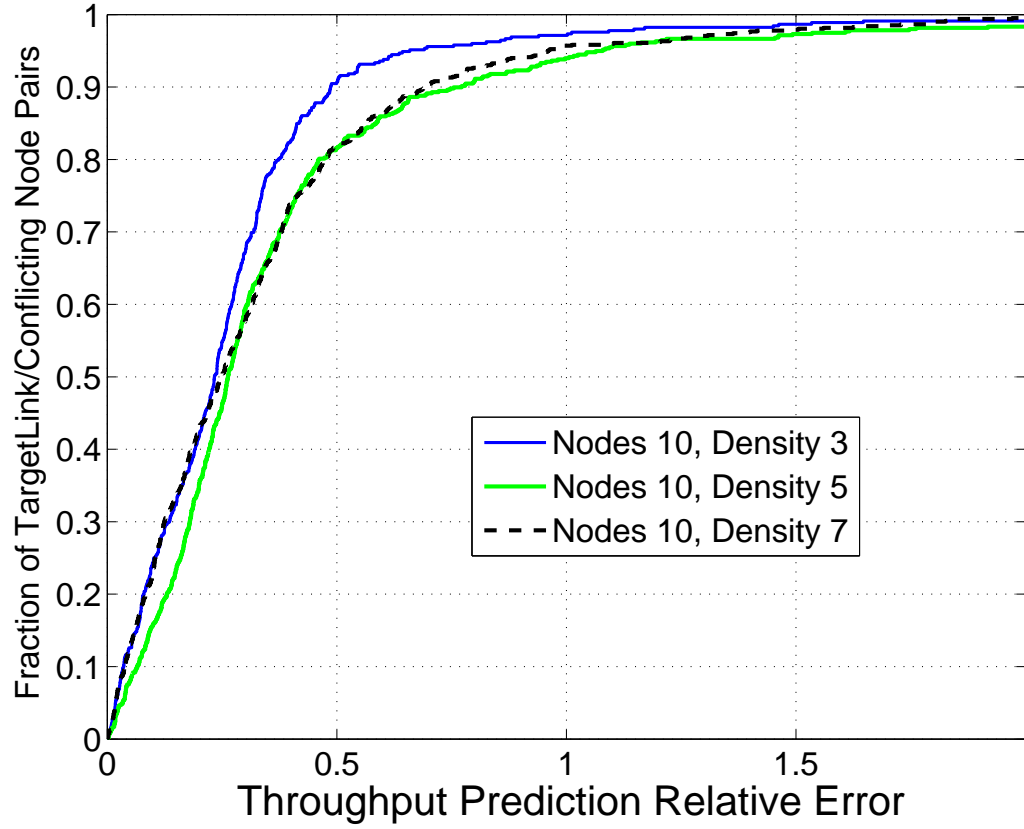


Figure 4.20 Throughput prediction sensitivity to density (900 kbps).

connections, as reported by WIANA [4], are depicted in Figure 4.21; the coverage area is $3km^2$, and the average node degree is 6.58. All nodes are equipped with 802.11b cards; the node denoted as *GW*, the gateway, is also wiredly connected to Internet via a 100 Mbps fiber. I considered a scenario of upload activity of the nodes, and the destination of each link is chosen according to the shortest path toward the gateway.

Inference Tool. Figure 4.22 compares the values of the Activity Share entries inferred by my solution with the actual values. All network nodes send fully backlogged

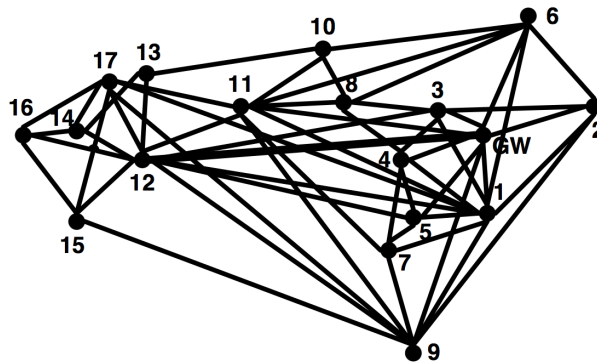


Figure 4.21 Topology of the TFA network deployed in Houston, Texas.

traffic at 11 Mbps (as shown above, this represents a worst case for the inference tool). The X-axis index identifies the network states, while the Y-axis represents the Activity Share, i.e., the ratio of time the network spent in a state. The X-axis is sorted according to increasing values of the actual Activity Share of the states and, for the sake of readability, represent only the 237 states with highest Activity Share. The graph shows again an excellent agreement among inference and simulation.

Throughput Prediction. Similarly to the case in Figures 4.19-4.20, I evaluate the accuracy of my methodology by predicting the throughput increase on any target link, achievable by limiting the transmission rate of any conflicting node; I consider all possible target link / conflicting node pairs. Figure 4.23 shows the empirical CDF of the absolute error between the throughput increase predicted by the methodology and the actual throughput increase. The X-axis represents the absolute error in terms

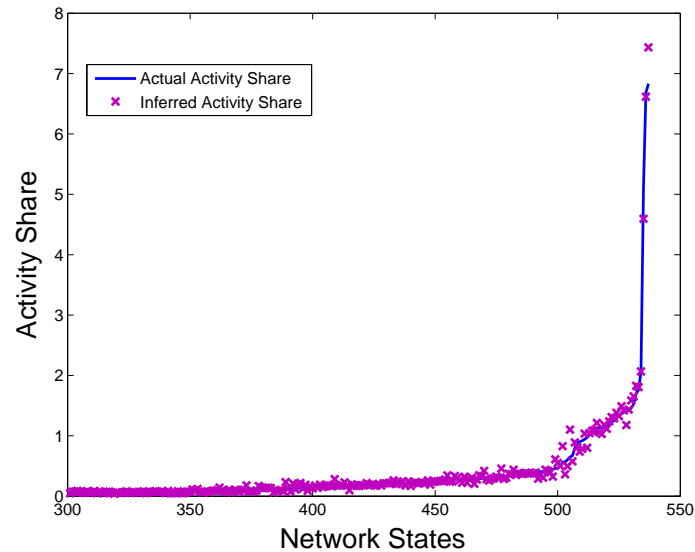


Figure 4.22 Activity Share Inference for the TFA network (fully backlogged nodes).

of packets per second, while the Y-axis represents the fraction of predictions. In this experiment, the target transmission rate is 600 kbps for all links excluding the target link, which is fully backlogged; the rate limitation is 50 packets per second (where the size of each packet is 1000 bytes). This permits to obtain appreciable throughput increases for several of the examined target link / conflicting node pairs. The figure shows that in 81% of the predictions the absolute error is less than 10 packets per second.

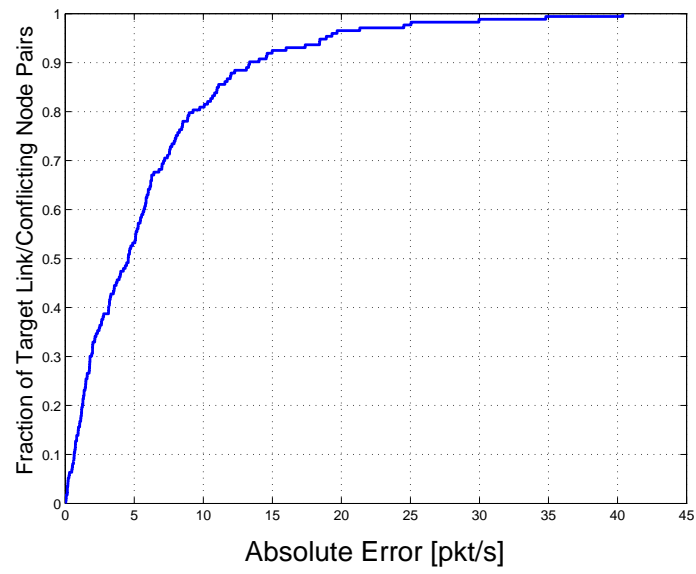


Figure 4.23 Throughput Increase Prediction for the TFA network (transmission rate 600 kbps).

Chapter 5

Related Work

5.1 WiFi-Nano

There has been tremendous amount of work targeted towards improving wireless performance. I discuss a few papers that are relevant to WiFi-Nano.

Performance. Researchers have proposed several techniques to improve performance in wireless LANs including use of partial packets in recovering from errors [27, 42], using network coding for retransmissions [57], using directional antennas [38, 43], and the use of aggregation and TDMA-like schedule to improve WiFi efficiency in the presence of VoIP traffic [69]. Perhaps closest to WiFi-Nano is Idle Sense [22] and FICA [64].

Idle Sense proposes an alternative to the binary exponential backoff-based WiFi MAC by trying to ensure that hosts in a single wireless LAN use a similar contention window. As I show in my evaluations, the low collision probability of Idle Sense helps WiFi-Nano use a shorter preamble length. FICA tackles the inefficiencies of the WiFi MAC by redesigning both the PHY/MAC using fine-grained *subchannels* and delivering efficiencies of 70% at 600 Mbps. However, the use of subchannels requires a synchronous system which makes it difficult for FICA to co-exist with neighboring networks that are not frequency/time synchronized.

Full-duplex. Recently, full-duplex single channel wireless communication sys-

tems have been proposed [13, 55]. The key challenge in these systems is eliminating the self-interference of the local transmitter, which can be done using a combination of analog interference cancellation and a nulling antenna. However, performing full-duplex decoding at MIMO data rates over 20-40 MHz bandwidth and at WiFi transmit power is still a challenging, open problem [13]. Finally, while these systems double the capacity of data transmission, they still suffer from the channel access overhead issues of WiFi.

In WiFi-Nano, I only need to correlate with the preamble during transmission and do not expect the transmitter to decode any bits. Thus, the interference cancellation requirement is not as stringent as full-duplex systems. Since WiFi-Nano reduces the channel access overhead, it is complementary to full duplex systems.

Collisions. When packet collisions occur in 802.11-based wireless networks, the receiving node may still be able to *capture* one of the transmissions if its signal strength is sufficiently high compared to the interfering signal, thereby reducing the impact of collisions [36]. However, the capture effect can result in unfairness where transmissions from nodes closer to the AP have higher probability of getting captured compared to nodes far away from the AP.

ZigZag decoding [21] is a technique that allows nodes to efficiently recover from collisions due to hidden terminals by exploiting interference-free stretches in the collided packets. CSMA/CN [60] performs collision notification where the receiver can

notify the transmitter to abort transmission when capture is not feasible during collisions, thereby reducing the collision overhead. WiFi-Nano also suffers from hidden terminal problems like WiFi and could benefit from CSMA/CN and ZigZag decoding to improve efficiency in the presence of hidden terminals. Further, the collision notification in CSMA/CN is achieved by correlating with a preamble-like sequence while packet transmission is on-going. WiFi-Nano’s speculative preamble relies on similar correlation ability at the transmitter except during preamble rather than packet transmission.

Finally, collisions due to overlapping channels is studied in [40] where the authors propose that retransmissions be permuted so that decoding efficiency is improved.

Measurements. Carrier sensing and packet detection algorithms are not specified in the standards and are left to the vendor’s implementation. Packet detection is accomplished through a combination of correlation with preamble and energy detection [26]. Different vendors have different thresholds for preamble and energy detection [39]. Similar to [39], I also incorporate preamble detection as part of carrier sensing in the Qualnet simulator for my evaluations.

The authors in [29] study the impact of interference on packet reception in 802.11b by carefully controlling the timing of the interferer with respect to the transmitter. They show that packet reception probability increases significantly as soon as the interferer is delayed from the start of the transmitter by as little as $3.2 \mu s$. While

802.11b preamble is DSSS (unlike the OFDM preambles used in 802.11g/n and in this paper), these results indicate the preamble detection can occur even in the presence of significant interference.

5.2 802.11ec

Many random access MAC protocols have been proposed to mitigate collisions and address hidden terminals, e.g., [5, 30, 66]. In contrast to tones and messages used by such protocols, I design a new primitive of correlatable symbol sequences which, by virtue of being short and robust, increases throughput and fairness with minimal overhead. More recently, while several papers have addressed 802.11 throughput overhead reduction [46, 61], they completely neglect the case of hidden terminals and, because of their more aggressive contention mechanisms, suffer severe throughput penalties in their presence. Ongoing standardization efforts, namely 802.11ah [2], target overhead reduction for sub-GHz communications, with application to smart grids, surveillance systems, etc. The strategy adopted in [2] consists in removing or compressing some information fields of the control messages, for a total of few bytes; compared to 11ec, this has a minor effect on the control message overhead, e.g., due to the preservation of the cumbersome preamble structure. Other techniques have been proposed to improve 802.11 throughput, e.g., [21, 41, 60], but address collision resolution rather than collision avoidance. For this reason, they are complementary to

(and can be used in combination with) 11ec. Furthermore, my physical layer model is significantly more simple than [21, 46, 60, 61], since it relies only on the replication of components (correlators) that are already present in common 802.11 cards. Finally, other applications of signal correlation have been recently shown in [23, 63, 73].

5.3 MIDAS

Wireless Network Monitoring. Performance monitoring of single-hop WLANs has recently attracted research interest [12, 47]. The proposed approaches reconstruct a global trace of all network packet transmissions by combining offline detailed traces reported by sniffers spread throughout the network. These solutions can provide a comprehensive survey of the network activity. However, they require the delivery of *detailed traces* from all (or at least most of) the nodes, which severely hinders the normal operations of multi-hop wireless networks. In my work, I show that I can attain very accurate results with the use of small time-averaged reports. Furthermore, [12, 47] do not address the problem of identifying the origins of poor link performance and rate-limiting the most hindering nodes.

802.11 Throughput Models. Several 802.11 throughput prediction models have been proposed in the literature [8, 11, 20, 28, 31, 49, 54, 70]. Their goal is either to compute the throughput of the network links given their traffic demands, or to compute the feasible region of the network. In contrast, I use measurements to

infer the network behavior, particularly the coordination between node transmissions and the causes of poorly performing links, and use this understanding to improve the throughput of under-served links. Our scheme relies on active offline link profiling, such as [31, 54], to identify the carrier sensing and interference relationships between the nodes. In addition, I introduce passive online measurements during normal network operations, to capture the complex node interactions determined by the actual transmission patterns. Recently, [32, 58] propose methods to avoid active offline profiling: [58] uses low overhead online probes; [32] uses passive online estimations using traces collected by deployed sniffers. While [32, 58] only consider pair-wise link interference and do not characterize the coordination between conflicting nodes, I can leverage the results therein for online link profiling.

Alternative Mitigation Techniques. Network managers have a number of options to optimize the performance of under-served links other than rate-limiting conflicting links. Alternative mitigation techniques have been suggested in literature, e.g., channel selection [33], association control [6], transmit power control [50], and channel width adaptation [51]. I remark that they are orthogonal to my inference technique and can benefit from it.

Chapter 6

Conclusions

This thesis addresses the problem of throughput loss and imbalance in wireless networks. Specifically, I focus on two fundamental causes, i.e., 802.11 protocol overhead and destructive interference. While addressing different aspects of the problem, two general principles emerge from the proposed approaches. The first principle is that increasing the physical layer data rates of wireless communications provides lesser and lesser returns, as the communication overhead and interference intrinsically limit the links' achievable throughput and cause imbalance. My research shows that the recent technological enhancements should be leveraged to, and have the potential of, holistically improving the effectiveness of MAC layer operations, i.e., including not only packet transmission but also, and in particular, the channel contention operations. The second principle is the importance of cross-layer considerations in the design of MAC layer solutions. This thesis shows that design interaction with lower, as well as with higher layers, are equally important in improving link performance. My research makes three main contributions.

WiFi-Nano. I identified the slot size as a key reason for 802.11 inability to deliver the data rate gains achieved at the physical layer to the MAC layer and above. WiFi-Nano reduces 802.11 9 μs slots to 800 ns slots. The key intuition

of WiFi-Nano is that, by parallelizing the historically serial operations, the former physical layer limits dictating the slot length can be overcome. Specifically, WiFi-Nano proposes speculative preamble transmissions, in which preamble transmission and detection are carried in parallel via self-interference cancellation. WiFi-Nano provides two key advantages: first, it implements a form of collision detection that nearly eliminates same slot collisions; second, it enhances transmission fairness due to the interaction between short slot size, propagation delay and heterogeneous SINRs via counter rollback. Using testbed experiments and extensive simulations, I show that WiFi-Nano is able to double the MAC throughput of WiFi at 802.11n rates.

802.11ec. 802.11ec is an 802.11-based protocol without control messages. 802.11ec's key contribution is the usage of correlatable symbol sequences to convey MAC control information; this provides robustness and efficiency to MAC control. Based on this technique, I design a complete protocol, whose main element is the mapping of control information to a limited number of CSS's and timing rules. Through a combination of experiments on a software defined radio, emulations, and simulations I show that 802.11ec improves network fairness by up to 90%, channel utilization by up to 46%, and throughput of under-served flows by over 22x, with respect to 802.11. .

MIDAS. MIDAS addresses the problem of identifying the conflicting nodes that cause underperformance of a target link. I introduce the key concept of Activity Share that captures the coordination among the conflicting nodes. Since the Activity

Share cannot be locally measured by the nodes, I show how MIDAS infers it using time-aggregate, passively collected measurements reported by the nodes. Finally, I design a throughput model based on the Activity Share that MIDAS utilizes to predict the benefit of rate-limiting conflicting transmissions. I implement MIDAS on an FPGA-based radio platform and show that it infers the Activity Share with a mean relative error as low as 4%, and predicts the throughput gain of an under-served link corresponding to alternative rate-limiting actions with an error lower than 20% of the rate-limited quantity.

Finally, the research discussed in this thesis has not only immediate application potentially influencing the standards, but also denotes principles opening novel paths that can be followed in several directions. My thesis points out several 802.11 inefficiencies and indicates that novel technologies, such as self-interference cancellation and signal correlation, provide adequate tools to highly reduce the hindering overhead, i.e., channel access and contention. While WiFi-Nano and 802.11ec show considerable gains over 802.11, there is still considerable room towards a vision of a random access overhead-free MAC. The combination of the two proposed solutions also represents a valuable contribution, whose challenges reside in designing novel preambles satisfying both protocol requirements, and in balancing the increased aggressiveness of WiFi-Nano in the presence of hidden terminals. Furthermore, MIDAS originally unveils the importance of transmission coordination to capture the network operating

point and characterize link interactions, including destructive interference. While in this thesis I show how to use this key insight to identify hindering transmissions, other applications including advanced examinations of link interactions can leverage it. Finally, differently from all related work, MIDAS proposes the first throughput model that is based on the current network state. This is a first step into a previously unexplored territory, which provides the opportunity to obtain more and more accurate predictions, by combining models and reality.

References

1. IEEE Std 802.11-2007 - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE*, 2007. <http://standards.ieee.org/about/get/802/802.11.html>
2. IEEE P802.11 Sub 1GHz Study Group. http://www.ieee802.org/11/Reports/tgah_update.htm
3. Rice University WARP project. Available at: <http://warp.rice.edu>
4. Wireless and Internet and Assigned and Numbers and Authority. <http://www.wiana.org>
5. V. Barghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: a Media Access Protocol for Wireless LAN's. In *Proc. of ACM SIGCOMM*, 1994.
6. Y. Bejerano, S. J. Han, and L. Li. Fairness and Load Balancing in Wireless LANs Using Association Control. In *Proc. of ACM MobiCom*, Sept. 2004, pp. 315–329.
7. G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 18(3): 535–547, March 2000.
8. R. R. Boorstyn, A. Kershenbaum, B. Maglaris, and V. Sahin. Throughput analysis in multihop CSMA packet networks. *IEEE Transactions on Communications*, 35(3):267–274, Mar. 1987.
9. J. Camp, V. Mancuso, O. Gurewitz, and E. Knightly. A measurement study of multiplicative overhead effects in wireless networks. In *Proc. of IEEE INFOCOM*, 2008.
10. J. Camp, E. Aryafar, and E. Knightly. Coupled 802.11 Flows in Urban Channels: Model and Experimental Evaluation. In *Proc. of IEEE INFOCOM*, 2010.
11. M. Carvalho and J. Garcia-Luna-Aceves. A scalable model for channel access protocols in multihop ad hoc networks. In *Proc. of ACM MobiCom*, 2004.
12. Y.-C. Cheng, J. Bellardo, P. Benkö, A. C. Snoeren, G. M. Voelker, and S. Savage. Jigsaw: Solving the puzzle of enterprise 802.11 analysis. In *Proc. of ACM SIGCOMM*, 2006.

13. J.I. Choi, M. Jain, K. Srinivasan, P. Levis, S. Katti. Achieving single channel, full duplex wireless communication. In *Proc. of ACM MobiCom*, 2010.
14. T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley, 1991.
15. T. Cui, L. Chen, and T. Ho. Energy Efficient Opportunistic Network Coding for Wireless Networks. In *Proce. of IEEE INFOCOM*, 2008.
16. D.S.J. De Couto, D. Aguayo, J.C. Bicket, R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of ACM MobiCom*, 2003.
17. R. Diestel. *Graph Theory (Graduate Texts in Mathematics)*. Springer, 2005.
18. E.O. Elliot. Estimates of Error Rates for Codes on Burst-Noise Channels. *Bell Syst. Tech. J.*, 42(5): 1977–1997, September 1963.
19. R. Gallager. *Discrete Stochastic Processes*. Kluwer, 1990.
20. M. Garetto, T. Salonidis, and E. W. Knightly. Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks. In *IEEE INFOCOM*, 2006.
21. S. Gollakota, and D. Katabi. Zig-zag Decoding: Combating Hidden Terminal in Wireless Networks. In *Proc. of ACM SIGCOMM*, 2008.
22. M. Heusse, F. Rousseau, R. Guillier, A. Duda. Idle sense: an optimal access method for high throughput and fairness in rate diverse wireless LANs. In *Proc. of ACM SIGCOMM*, 2005.
23. S. Hong, S. Katti. DOF: A Local Wireless Information Plane. In *Proc. of ACM SIGCOMM*, 2011.
24. R.K. Jain, D.M.W. Chiu, W. Hawe. A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems. *Technical Report TR-301*, DEC Research, 1984.
25. K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *Proc. of ACM MobiCom*, 2003.
26. K. Jamieson, B. Hull, A. Miu, and H. Balakrishnan. Understanding the real-world performance of carrier sense. In *Prof. of ACM E-WIND*, 2005.
27. K. Jamieson, and H. Balakrishnan. PPR: Partial packet recovery for wireless networks. In *Prof. of ACM SIGCOMM*, 2007.

28. A. Jindal and K. Psounis. Characterizing the achievable rate region of wireless multi-hop networks with 802.11 scheduling. *ACM/IEEE Transactions on Networking*, 18(3):257–281, Mar. 2009.
29. G. Judd, and P. Steenkiste. Characterizing 802.11 wireless link behavior. *Springer Wireless Networks*, 16(1):167–182, Jan. 2010.
30. P. Karn. MACA- A New Channel Access Method for Packet Radio. In *Proc. of ARRL Computer Networking Conference*, 1990.
31. A. Kashyap, S. Ganguly, and S. R. Das. A measurement-based approach to modeling link capacity in 802.11-based wireless networks. In *Proc. of ACM MobiCom*, 2007.
32. A. Kashyap, U. Paul, and S. R. Das. Deconstructing interference relations in WiFi networks. In *Proc. of IEEE Secon*, 2010.
33. B. Kauffman, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki, and C. Diot. Measurement-Based Self Organization of Interfering 802.11 Wireless Access Networks. In *Proc. of IEEE INFOCOM*, May 2007, pp. 1451–1459.
34. S.M. Kay. *Fundamentals of Statistical Signal Processing - Vol. 2*. Prentice Hall, 1998.
35. F.P. Kelly, A.K. Maulloo, D.K.H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3): 237–252, March 1998.
36. A. Kochut, A. Vasan, A.U. Shankar, A. Agrawala. Sniffing out the correct physical layer capture model in 802.11 b. In *Proc. of IEEE ICNP*, 2004.
37. K. LaCurts, H. Balakrishnan. Measurement and Analysis of Real-World 802.11 Mesh Networks. In *Proc. of ACM IMC*, 2010.
38. S. Lakshmanan, K. Sundaresan, S. Rangarajan, R. Sivakumar. The myth of spatial reuse with directional antennas in indoor wireless networks. In *Proc. of Springer PAM*, 2010
39. J. Lee, J. Ryu, S.J. Lee, T.T. Kwon. Improved modeling of IEEE 802.11 a PHY through fine-grained measurements. *Elsevier Computer Networks*, 54(4): 641–657, April 2010.

40. L.E. Li, K. Tan, H. Viswanathan, Y. Xu, Y.R. Yang. Retransmission repeat: simple retransmission permutation can resolve overlapping channel collisions. In *Proc. of ACM MobiCom*, 2010.
41. T. Li, M.K. Han, A. Bhartia, L. Qiu, E. Rozner, and Y. Zhang. CRMA: Collision-Resistant Multiple Access. In *Proc. of ACM MobiCom*, 2011.
42. K.-C.J. Lin, N. Kushman, and D. Katabi. ZipTx: Harnessing partial packets in 802.11 networks. In *Proc. of ACM MobiCom*, 2008.
43. X. Liu, A. Sheth, M. Kaminsky, K. Papagiannaki, S. Seshan, and P. Steenkiste. Pushing the envelope of indoor wireless spatial reuse using directional access points and clients. In *Proc. of ACM MobiCom*, 2010.
44. E. Magistretti, O. Gurewitz, and E. W. Knightly. Inferring and mitigating hindering transmissions in managed 802.11 wireless networks. In *Proc. of ACM MobiCom*, 2010.
45. E. Magistretti, O. Gurewitz, and E. W. Knightly. Inferring and mitigating hindering transmissions in managed 802.11 wireless networks. In *Rice University Technical Report*, 2010. TREE3610. Available at: <http://www.ece.rice.edu/~knightly/Infer.pdf>.
46. E. Magistretti, K.K. Chintalapudi, B. Radunovic, and R. Ramjee. WiFi-Nano: Reclaiming WiFi Efficiency via 800 ns Slots. In *Proc. of ACM MobiCom*, 2011.
47. R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Analyzing the MAC-level behavior of wireless networks in the wild. In *Proc. ACM SIGCOMM*, 2006.
48. B. McFarland, A. Shor, and A. Tabatabaei. A 2.4 & 5 GHz dual band 802.11 WLAN supporting data rates to 108 Mb/s. In *Annual Technical Digest of IEEE GaAs IC*, 2002.
49. K. Medepalli and F. Tobagi. Towards performance modeling of IEEE 802.11 based wireless networks: A unified framework and its applications. In *Proc. of IEEE INFOCOM*, 2006.
50. V. Mhatre, K. Papagiannaki, and F. Baccelli. Interference mitigation through power control in high density 802.11 WLANs. In *Proc. of IEEE INFOCOM*, May 2007, pp. 535–543.
51. T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, P. Bahl, and Y. Yuan. Load-Aware Spectrum Distribution in Wireless LANs. In *Proc. of IEEE ICNP*, 2008.

52. S. Nedeveschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *Proc. of USENIX NSDI*, 2008.
53. D. Niculescu. Interference map for 802.11 networks. In *Proc. of ACM IMC*, 2007.
54. L. Qiu, Y. Zhang, F. Wang, M. K. Han, and R. Mahajan. A general model of wireless interference. In *Proc. of ACM MobiCom*, 2007.
55. B. Radunovic, D. Gunawardena, P. Key, A. Proutiere, N. Singh, V. Balan, G. Dejean. Rethinking Indoor Wireless Mesh Design: Low Power, Low Frequency, Full-duplex. In *Proc. of IEEE WiMesh*, 2010.
56. M. Richards. *Fundamentals of Radar Signal Processing*. McGraw-Hill, 2005.
57. E. Rozner, A.P. Iyer, Y. Mehta, L. Qiu, and M. Jafry. In *Proc. of ACM CoNEXT*, 2007.
58. T. Salonidis, G. Sotiropoulos, R. Guerin, and R. Govindan. Online Optimization of 802.11 Mesh Networks. In *Proc. of ACM CoNext*, Dec. 2009, pp. 61–72.
59. D.V. Sarwate, and M.B. Pursley. Crosscorrelation properties of pseudorandom and related sequences. *Proceedings of the IEEE*, 68(5): 593–619, May 1980.
60. S. Sen, R.R. Choudury, and S. Nelakuditi. CSMA/CN: Carrier Sense Multiple Access with Collision Notification. *Proc. of ACM MobiCom*, 2010.
61. S. Sen, R.R. Choudury, and S. Nelakuditi. No Time to Countdown: Moving Backoff to the Frequency Domain. In *Proc. of ACM MobiCom*, 2011.
62. T. Schmidl, and D. Cox. Robust frequency and timing synchronization for OFDM. *IEEE Transactions on Communications*, 45(12), 1613–1621, December 1997.
63. K. Tan, H. Liu, J. Fang, W. Wang, J. Zhang, M. Chen, and G. Voelker. SAM: Enabling Practical Spatial Multiple Access in Wireless LAN. In *Proc. of ACM MobiCom*, 2009.
64. K. Tan, J. Fang, Y. Zhang, S. Chen, L. Shi, J. Zhang, and Y. Zhang. Fine-grained channel access in wireless LAN. In *Proc. of ACM SIGCOMM*, 2010.
65. J. Thompson, B. Baas, E.M. Cooper, J.M. Gilbert, G. Hsieh, P. Husted, A. Lokanathan, J. S. Kuskin, D. McCracken, B. McFarland, T.H. Meng, D. Nakahira, S. Ng, M. Rattehalli, J.L. Smith, R. Subramanian, L. Thon, Y.-H.

- Wang, R. Yu, and X. Zhang. An Integrated 802.11a Baseband and MAC Processor. In *Proc. of IEEE ISSCC*, 2002.
66. F. Tobagi, and L. Kleinrock. Packet Switching in Radio Channels: Part II—The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution. *IEEE Transactions on Communications*, 23(12): 1417–1433, December 1975.
 67. F. Tufvesson, O. Edfors, and M. Faulkner. Time and frequency synchronization for OFDM using PN-sequence preambles. In *Proc. of IEEE VTC*, 1999.
 68. R. Van Nee, and R. Prasad. *OFDM for Wireless Multimedia Communications*. Artech House, 2000.
 69. P. Verkaik, Y. Agarwal, R. Gupta, and A. Snoeren. Softspeak: making VoIP play well in existing 802.11 deployments. In *Proc. of USENIX NSDI*, 2009.
 70. X. Wang, and K. Kar. Throughput modelling and fairness issues in CSMA/CA based ad-hoc networks. In *Proc. of IEEE INFOCOM*, Mar. 2005, pp. 23–34.
 71. A. Willig, M. Kubish, C. Hoene, A. Wolisz. Measurements of a Wireless Link in an Industrial Environment Using an IEEE 802.11-Compliant Physical Layer. *IEEE Trans. on Industrial Electronics*, 49(6): 1265–1282, December 2002.
 72. X. Zeng, R. Bagrodia, M. Gerla. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *Proc. of IEEE PADS*, 1998.
 73. X. Zhang, K. Shin. E-MiLi: Energy-Minimizing Idle Listening in Wireless Networks. In *Proc. of ACM MobiCom*, 2011.